

2022 必致(BizDevOps) 白皮书

业务研发运营一体化模型及
实践框架



数字化转型
使命必达

业技融合
行稳致远

编写组

BizDevOps共促计划专家组

编写组成员名单

何 勉 肖 然

张 裕 陈展文

张燎原 张 贺

欧 红 付晓岩

陈 鑫 亢江妹

BizDevOps共促计划成员单位

南京大学软件研发效能实验室

思特沃克软件技术(北京)有限公司

阿里云计算有限公司

北京极客邦科技有限公司

招商银行股份有限公司

上海优川信息技术有限公司

白皮书版权所有

BizDevOps共促计划全体单位



Contents

序

----- 1

前言

必致(BizDevOps)
打造数字化时代的高绩效组织

----- 3

01

必致(BizDevOps)的目标、 能力和实践

- 1.1 BizDevOps的1个总体目标 ----- 5
- 1.2 BizDevOps的3个能力要求 ----- 6
- 1.3 BizDevOps的5个关键实践 ----- 7

02

必致(BizDevOps)概念模型

2.1 定义概念模型,推动BizDevOps落地	8
2.2 概念模型的构建	9
2.3 模型的阅读和使用说明	14

04

必致(BizDevOps)实践案例

4.1 案例一:迈向BizDevOps的 招商银行精益管理体系	37
4.2 案例二:阿里巴巴,以应用为 核心打造持续业务交付能力	41

03

必致(BizDevOps)实践体系

3.1 BizDevOps实践的总体介绍	16
3.2 协作和管理领域的实践	17
3.3 工程和技术领域的实践	25
3.3 度量和改进实践	31

附录

附录一:BizDevOps共促计划

	47
--	----

附录二:参考资料

	47
--	----

附录三:词汇表

	48
--	----

序

2012年全球最具影响力的研究咨询机构Forrester曾预言：

“In the future, all companies will be software companies”(在未来,所有的企业都将成为软件企业)

10年前,我还在澳洲和欧洲从事软件工程的科研与实践工作,这个预言对当时的我产生的震撼是无比强烈的。过去的10年间,我们见证着全社会数字化的进程,同时也见证着这一预言正在逐渐变成我们身边的现实。现在我们能够充分地理解,为了将业务接入到正在全面数字化的社会,每一个企业和机构都必将运行在软件之上。

半年多之前,当BizDevOps共促计划的发起人何勉老师联系我,相邀一起在国内提出BizDevOps倡议,我感到非常兴奋并欣然应允。BizDevOps是DevOps在数字经济的根本性拓展和变革,这与我所领导的南京大学软件研发效能实验室一直以来所倡导的DevOps+的理念不谋而合,也是近年来国际软件工程学术界兴起的持续软件工程概念的重要组成部分。近10年来,DevOps运动在全球和中国风起云涌,已成为软件产业先进生产力的代表。然而,DevOps将关注点主要放在打破开发与运维之间的壁垒,虽然极大地提升了软件的研发运维效率,但尚未形成完整的价值闭环。BizDevOps倡导开发与运维之间的整合向前进一步扩展延伸到业务,使能业务作为价值的起点及核心目标,充分、高效地对接到DevOps的价值实现引擎。

BizDevOps的愿景十分美好,它要求各职能围绕业务价值进行高效协同,特别是要打通业务到开发运维的端到端价值交付链路,形成反馈闭环。但在实践中要切实打破业务与研发之间的屏障,使业务人员与研发人员能够无缝、顺畅地高效协同,面临着组织、流程、技术等层面的诸多挑战。诸如,业务与研发的目标不一致,各职能之间缺乏统一的沟通语言,工程与业务的实践彼此脱节、工具平台难以整合,各环节数据相对封闭、不能有效关联等等。为了指导企业向BizDevOps的成功变革,《BizDevOps白皮书》提出1个目标、3个能力和5个关键实践,以厘清对BizDevOps的明确定义,达成共同认知;在此基础上,建立起BizDevOps的概念模型和实践体系,为国内企业和实践者勾绘出BizDevOps方向的第一张实施路线图。

在Forrester预言的大时代背景下,相信BizDevOps理念和实践的导入和落地必将助力每一个现在或未来运行于软件之上的企业,加速价值流动的效率和价值交付的质量。在BizDevOps的牵引下,DevOps将不单作为高效价值实现的引擎,而终将演进成为企业的创新实验平台和业务发展引擎,真正实现从成本中心向利润中心,再向创新中心的跃迁。

张贺

南京大学教授,博士生导师,软件研发效能实验室主任

2022年12月于南大北园

前言

必致(BizDevOps)——打造数字化时代的高绩效组织

随着云原生、元宇宙、Web3等技术拉开序幕，智能制造、智慧城市、精准医疗等应用场景徐徐展开，继人类工业文明之后，下一个大变局的奇点临近。毫无疑问，以数字技术应用为主线的数字化转型是此次人类文明变革的核心动力。在这一变革过程中，技术与业务的关系正发生根本性的转变，技术开发和交付方式也随之升级。

下图回顾了中国改革开放40多年所经历的主要阶段，业务与技术之间关系的转变，以及这一关系所催生的技术交付模式的变革。

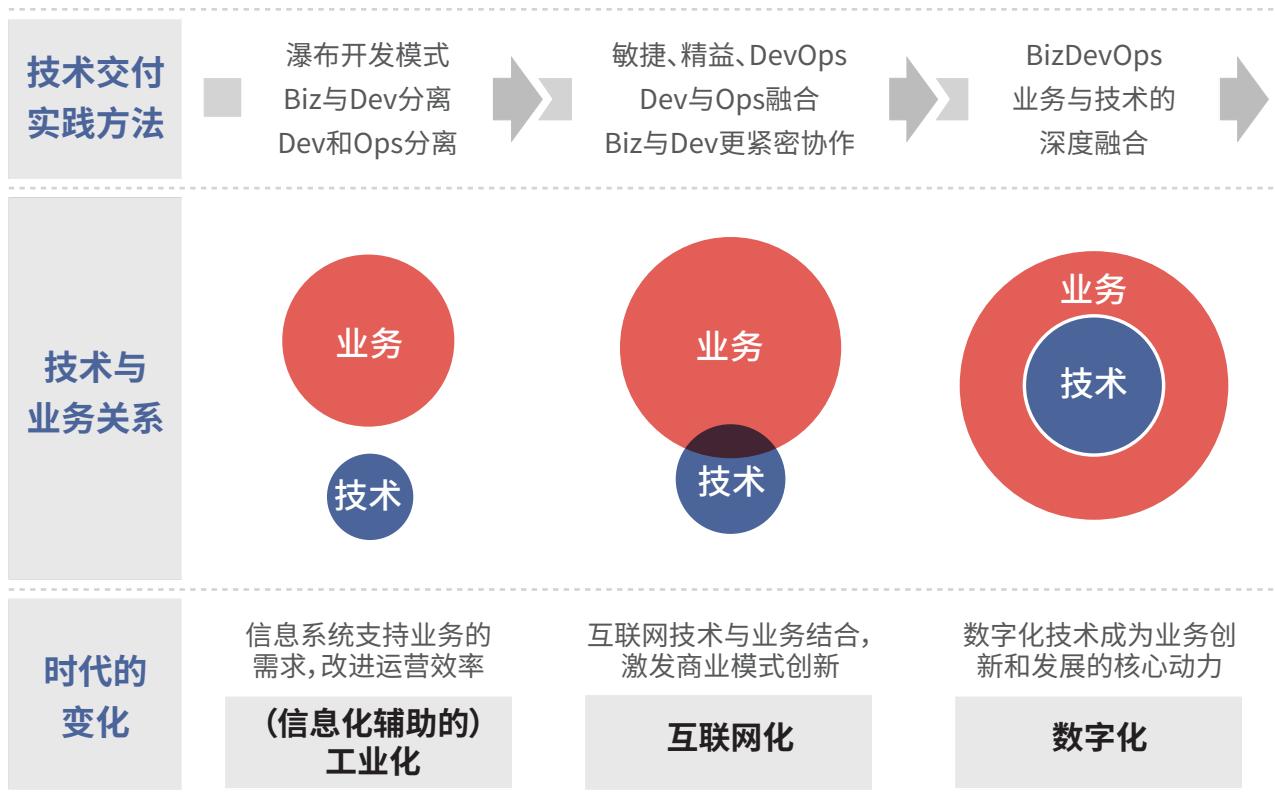


图0-1:从信息化到互联网经济,再到数字经济时代,技术和业务的关系在持续变化,IT开发和交付方式也随之演进

• (信息化辅助的) 工业化阶段

工业化的进程让中国制造从作坊变成了高度自动化的流水线。信息技术在其中起到了推动和支撑作用, OA(办公自动化)、ERP(企业资源规划)、金融信息系统, 以及各类MIS(管理信息系统)让先进的管理理念和规则得以落地, 让各个环节的运营效率成倍提升, 成就了具备国际竞争力的产业集群。

在工业化阶段, 技术(主要是IT技术)与业务相对独立, 并支撑业务。如图中所示, 此时IT技术与业务还是相对独立的。与之对应, IT开发与交付的特征是: 业务与开发相对独立——分离的技术团队接受来自业务的明确需求; 开发与运维相对独立——开发完成后, 由运维团队集中统一部署实施。

相对“软件作坊”, 瀑布模式代表了先进生产方式, 是工业时代开发和交付的主流模式。瀑布模式强调各个阶段(如: 需求、开发、测试、部署)界限和职责分明, 寻求确定的计划和执行。在工业化阶段, 这是合理和可行的, 而信息化也为工业时代提供了极大助力。

• 互联网阶段

上世纪末开始, 互联网技术的应用首先带来服务业全面升级, 从娱乐、咨询, 到购物、金融、再到教育、政务, 都发生了颠覆式的变化。如图中所示, 此时的技术与业务开始相互结合, 并催生了电商、互联网金融、生活服务等新的商业模式。互联网成为社会经济的重要组成部分。

在互联网经济下, 技术成为业务发展的重要变量, 业务的不确定性大为提升。业务与技术的界限开始模糊, 小步快跑、快速迭代、反馈试错等成为业务和技术的共同追求。此时, 敏捷和精益开发模式登上舞台, 逐渐占据主流, 强调业务与技术更紧密协作的精益创业理念也被广泛接受。

在这一阶段的IT开发和交付方法的集大成者非DevOps(开发运维一体化)莫属。它以系统的实践打破开发和运维的边界, 构建起

快速开发、交付和反馈闭环, 大大加速了从产品想法到客户反馈的闭环。DevOps运动中, 卓越互联网产品团队一天可以完成几十上百次这样的验证闭环。DevOps指引了互联网时代IT开发和交付方法的演进方向, 也奠定了数字化时代技术交付方法的基础。

• 数字化阶段

时代的科技车轮加速前进, 站在当下的数字化时代, 数字经济正沿着产业数字化和数字产业化这两大主题高速演进。一方面, 技术的前沿不断向前推进: 从IoT到机器人, 从区块链到Web3, 从云原生到量子计算, 从5G到边缘计算, 从虚拟现实到元宇宙, 让人应接不暇; 另一方面, 数字技术加速融入并深刻改变每一个产业, 各行各业的价值链被数字技术重塑, 数字化转型成为各行各业的共识。

应用数字技术, 重塑业务价值链, 提升价值交付的效率、质量和体验, 这是数字化转型的本质。数字化时代, 一切业务都将运行在数字化技术之上。正如上图所示, 技术将成为业务的内核, 业务的进化与技术的进化融为一体。这令人兴奋, 同时也带来了组织的集体焦虑——如何保证数字化转型资源投入的有效性成为组织发展的核心命题。

在产业数字化的趋势下, 技术成为业务演进和创新的内核。业务和技术的合作关系也急需升级, 突破点在于打造业务与技术深度融合的组织机制与实践方法, 而这一方法就是BizDevOps。

我们将BizDevOps的中文称为“必致”, 一方面是“Biz”的音译, 体现了服务业务数字化转型这一核心使命; 另一方面, 我们相信只有通过业务与技术的有机融合, 才能够稳步达成这一使命。这两者加起来就是我们的长期愿景——数字化转型使命必达, 业技融合行稳致远。“必致”两个字分别取自这两句话。

发布**必致(BizDevOps)白皮书**是为了顺应数字化时代的要求, 总结行业的先进理念和实践, 推动BizDevOps理念方法的普及, 以及实践的应用和落地, 从而打造数字化时代的高绩效组织。

必致(BizDevOps)的目标、能力和实践

特斯拉的创始人埃隆·马斯克在一次采访中说道：“设计生产机器(汽车)的机器(汽车生产流水线)，比制造机器本身要困难十倍、百倍。而人们经常认识不到这一点”。

相对特定的数字业务，更困难和重要的是数字业务创新和发展的机制方法。构建制造机器的机器是企业数字化转型的头等大事。前一个(被制造的)机器指的是数字业务本身，而第二个(制造机器的)机器就是BizDevOps。

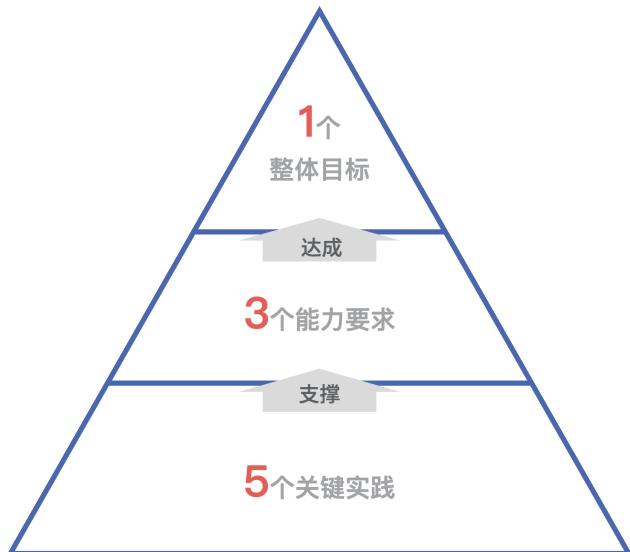


图1-1: BizDevOps的实践、能力、目标

我们将从目标、能力和实践这三个方面，完整定义BizDevOps。如上图所示，我们将其总结为1个总体目标，3个能力要求和5个关键实践。

BizDevOps的1个总体目标

BizDevOps的总体目标是：**打造业务和技术有机融合、高效运作的数字化组织，赋能数字业务的持续创新和长期发展。**

BizDevOps是企业数字化转型的重要组成部分——为业务的数字化转型打造数字化的组织。尽管DevOps是BizDevOps实践组成部分，但BizDevOps并不是DevOps2.0，而是以业务为核心构建的整体体系。业务是BizDevOps的起点，是贯穿其始终的核心要素，更是它的最终目标。

“

成功的BizDevOps实施是组织由茧化蝶的过程，而蝶变的目标是打造服务数字化业务的数字化组织。离开了业务这一核心，即使其它实践再完美，你所拥有最多不过是一条爬得更快的毛毛虫。

”

与BizDevOps的目标对应，本白皮书的核心也是围绕业务展开，着重于解决业务与技术融合的模型和实践方法。DevOps中已经涵盖的产品研发内部以及产品研发与运维的协同实践，不是本白皮书的重点，除非它与业技融合相关。

BizDevOps的3个能力要求

为了实现BizDevOps的总体目标,组织需要建立三个方面的能力,它们分别是:

1. 以客户价值为核心的协同能力

BizDevOps服务于数字业务的创新与发展,它要求组织的各个职能围绕客户价值高效协同。为此,组织必须打通从业务(Biz)到产品开发(Dev)到系统运维和运营(Ops)的端到端价值交付链,并形成有效的反馈、调整闭环。

打通Biz, Dev和Ops的链路,也是BizDevOps名称的由来。其中的Ops包括系统运维,更包括业务运营,BizDevOps要建立的是从业务开始到业务结束的完整链路和反馈闭环。

2. 全链路的数字化运作能力

客户价值为核心的协同能力是BizDevOps的基础,但它也提高了协作的复杂度和能力要求。组织必须建立全链路的数字化运作,才可能做到有效的协同。这里的全链路指的是从业务价值的发掘、分析、规划到交付、运营、反馈、调整的完整链路。为实现全链路数字化运作,各个环节的运作必须建立在统一的模型基础之上,这样才能从底层链接各个环节,实现跨环节的数据共享,并形成高效运作和有效反馈的机制。

数字化的运作,一方面为协同和交付的质量、效率和有效性提供了保障;另一方面,基于统一模型的数字化运作体系,可以产生高可用的数据,它具备三个特征:

全量

运作过程的所有操作数据被记录

全要素

基于一致的模型,让数据拥有所需要的属性描述,且数据间可以被正确关联、完整刻画

实时

数据在操作过程的第一现场和第一时间产生,既保证其真实性,又保证其即时可用

3. 基于高可用数据的过程透明和效能度量能力

BizDevOps体系的建设是一个持续的过程,一方面需要保障其落地执行,另一方面更需要持续改进。

全链路的数字化运作,产出了高可用的数据。基于高可用的数据,组织需要建立透明交付过程,和度量组织效能能力。交付过程的透明,可以进一步保障价值交付过程的执行;面向场景目标的度量,可以引导持续的效能改进。



图1-2: BizDevOps的3个能力要求

如上图所示,以上三个能力构成一个完整的循环——以客户价值为核心组织协同,打通了从Biz到Dev到Ops的交付链路和反馈闭环;在此基础之上的全链路数字化运作能力,则保障系统交付的效率、质量和有效性,并产生高可用的数据;基于高可用的数据,透明过程并度量效能,则保障数字化运作的有效执行,并持续提升组织效能。最终实现真正意义上的以客户价值为核心的高效协同、有效创新。

BizDevOps的5个关键实践

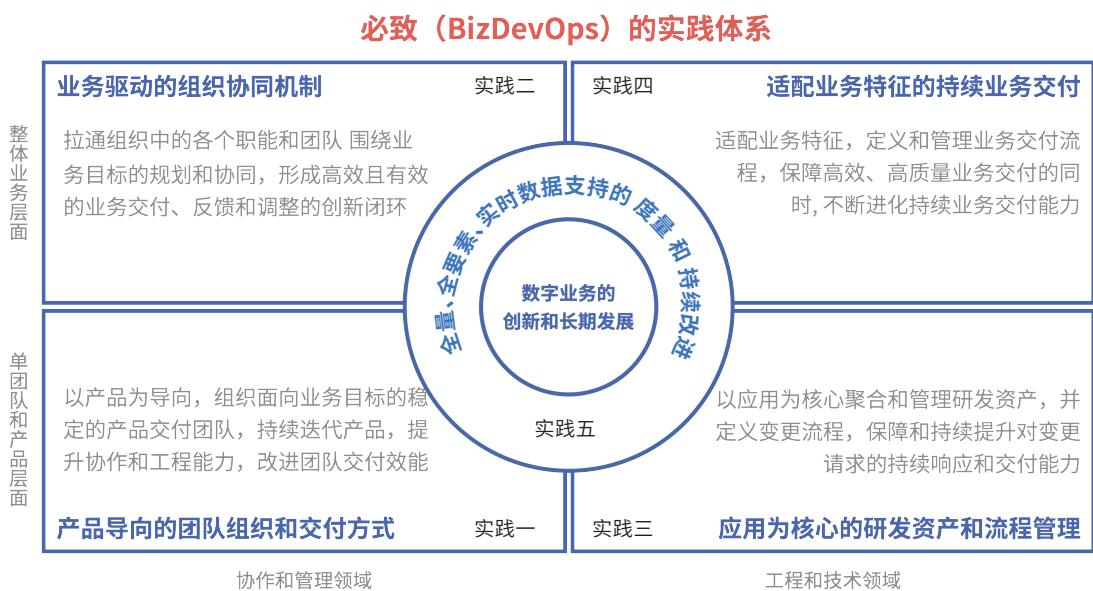


图1-3: BizDevOps的5个关键实践

BizDevOps的目标和能力,需要实践支持才能达成。我们总结了不同企业的成功经验,汇总并定义了BizDevOps的五个关键实践。

如上图所示,BizDevOps实践体系,按上下和左右分成4个实践,其中:上半部分关注整体业务层面,下半部分关注独立的团队或产品层面;左半部是协作和管理领域的实践,右半部分是技术和工程领域实践。再加上中间的与前4大实践正交的度量和持续改进实践,共5个实践。它们分别是:

- 实践一** 产品导向的团队组织方式
- 实践二** 业务驱动的组织协同机制
- 实践三** 应用为核心的研发资产和流程管理
- 实践四** 适配业务特征的持续业务交付
- 实践五** 全量、全要素、实时数据支持的度量和持续改进

它们构成了业务技术融合的实践体系,帮助组织实现前文中所述的三个能力要求,并赋能数字业务的持续创新和长期发展这一总体目标。白皮书第三部分,将分别详细介绍这5项实践。

上面,我们介绍了BizDevOps的1个总体目标,3个能力要求和5个关键实践,我们称其为“BizDevOps的1,3,5”。综上所述,我们可以完整的定义BizDevOps:

BizDevOps是数字化时代的业务创新及产品交付实践和方法体系,它通过业务和技术有机融合和有效协同,打通从业务到开发再到运维和运营的价值交付链路和反馈调整闭环,并实现全链路数字化运作,保障和持续改进产品和业务交付的效率、质量和有效性,从而赋能数字业务的持续创新和长期发展。

必致(BizDevOps) 概念模型

／ 定义概念模型, 推动BizDevOps落地

在业务数字化的浪潮下,组织本身的数字化变得越发重要。BizDevOps打造的正是服务数字业务的数字化组织(Digital Organization for Digital Businesses)。多数企业的管理层对建设业务技术融合的数字化组织是非常支持和期待的。然而,到了规划和实施层面,从哪里开始实施BizDevOps,以及怎样才能融合业务与技术却面临很多挑战。以下总结了其中常见的困难:

缺乏统一目标

技术与业务的目标不一致,很多时候目标定义的出发点和维度很难统一

话语体系不同

部门间缺乏统一的话语体系,沟通困难,很难形成有效结论和可落地的行动

实践相互脱节

业务和技术的实践相互脱节,管理实践(如需求的管理和迭代计划)和工程实践(如系统集成、部署和发布)相互脱节,造成价值交付链路不顺畅

工具平台割裂

业务和技术有各自的工具平台,但很难有效地集成互通

数据孤岛化

各个环节的数据不能有效关联,可用性差,很难发挥数据的生产要素作用

做不到持续进化

实施中形成的组织、流程和工程资产缺乏结构化的组织,很难有效的沉淀和持续的进化,流程机制和工具平台经常都面临要么僵化不前、要么推倒重来的困境

这些挑战尽管表象不同,背后却有共同的原因,那就是:对BizDevOps的问题域缺乏一致认知和清晰的定义。问题域是相对解决方案域而言,它指对要解决的问题的系统定义,包括:问题是什么,涉及哪些概念,这些概念之间的关系是什么。

爱因斯坦曾经说过:“如果我有1小時拯救世界,我会花55分钟去定义问题,再花5分钟找到解决方案”。如果问题域定义不清,就无法产出可落地和可持续的解决方案。同样,如果对问题域的认知不一致,就无法做到高效沟通,也无法统一目标。

BizDevOps共促计划成立后,专家委员会一致认为问题域定义十分关键,并把BizDevOps概念模型作为定义BizDevOps问题域的载体。因此,共促计划把定义BizDevOps概念模型当做了专家委员会的第一项重要工作,进行了多次线上和线下研讨,梳理、总结和抽象BizDevOps的核心概念模型,并把它作为统一认知和指导实践的依据。

本着开放普惠的科技精神,本白皮书将第一次公布阶段性成果——BizDevOps概念模型的预览版本,今后我们还会定期更新模型以反映行业的最新进展,以及我们认知的进化。

概念模型的构建

BizDevOps概念模型必须支持主要的价值交付链路,如需求的交付、缺陷的处理等,这样模型才可能实用和完备。因此,我们决定从识别价值交付链路开始,逐步构建BizDevOps概念模型。

理解构建过程,可以帮助大家理解模型的本质,并在实际场景中更好地应用它。接下来,我们将通过还原模型的动态构建过程来介绍整个模型。

• 步骤1:识别主要价值交付链路及它的作业对象

构建模型的第一步是识别BizDevOps的主要价值交付链路。

价值交付链路对应一类价值的完整实现过程,例如:业务的交付和反馈就是一个典型的价值交付链路。它包含从收集或产生业务需求,到分析、规划、实现,再到验收、发布,直至反馈调整的整个过程。

本质上,每个价值交付链路都是一个作业过程,被交付的价值

单元是它的作业对象。在上面的例子中,业务的交付和反馈链路的作业对象是业务需求。作业对象随作业过程发生状态迁移和流转,最终实现价值的交付。

下图表达了BizDevOps最核心的价值交付链路,其中,橙色矩形框代表的是该链路的作业对象。各个价值交付链路之间可能会存在关联,如:业务交付和反馈链路会派生出产品交付链路。价值链路间的关联会体现到作业对象之间。上例中对应的就是业务需求分解为1到多个产品功能需求。图中作业对象间的连线表示这种关联。BizDevOps典型的价值交付链路可分为两大类。

第一是协作类的,除业务交付和反馈链路外还包括:产品设计和交付链路以及缺陷的处理链路等。其中,产品设计和交付链路的作业对象是产品功能需求;缺陷的处理链路的作业对象是缺陷。

第二是工程类的,包括:应用的变更链路和系统或版本的发布链路等。其中应用的变更指的是针对某个产品功能而要做的技术改动的交付过程,包括:设计、编码、提交和部署的过程,它的作业对象是变更请求(Change Request);系统或版本的发布,指对外的一次需求或版本的发布,它的作业对象是发布(Release)。

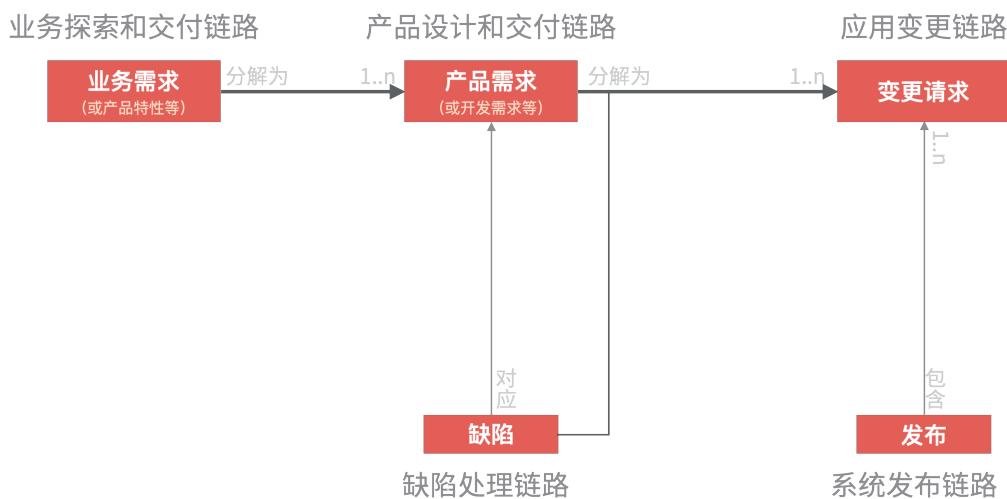


图2-1:BizDevOps模型的核心——各个价值交付链路上的核心作业对象

需要说明的是,以上作业对象的命名,选取了最典型的命名。在现实中,因具体企业或业务特征的不同,命名可能不同。如,在标准产品售卖为商业模式的组织中,业务需求更可能被产品特性取代,变更请求在不同组织中可能被称为变更、变更单或技术任务等。产品功能需求,可能被称为开发需求或Story等。本白皮书对作业对象的命名,在实际应用中可以也需要被适配。

相互关联的作业对象是BizDevOps概念模型的内核,模型将围绕这一内核展开,而检验模型设计完整性和有效性的基本标准是:能否高效支撑各个作业链路的有效运作。在这一前提下,模型应该尽可能简洁明了。

接下来,将从作业链路和作业对象出发,逐步构建BizDevOps概念模型。为了避免重复,在后面的模型中将省略价值交付链

路的名称,它们已经蕴含在作业对象当中了。

• 步骤2:为各个流程确定作业空间,并划分模型的领域

作业要在特定空间内完成,我们称其为作业空间。

作业空间是一类或多类角色为了共同的目的,作业和协作的功能区域。作业空间提供特定的功能集合,可以进行配置、数据和权限的管理,同时还应该提供或链接完成作业所需要的资源,存放或链接作业过程和最终的产物。

如下图所示,蓝色的矩形框是作业空间,包括:产品线、交付团队、应用、发布单元等,作业对象隶属于这些空间,在其中完成作业。下表列出了各作业空间,它们管理的作业对象以及具体作业链路等。

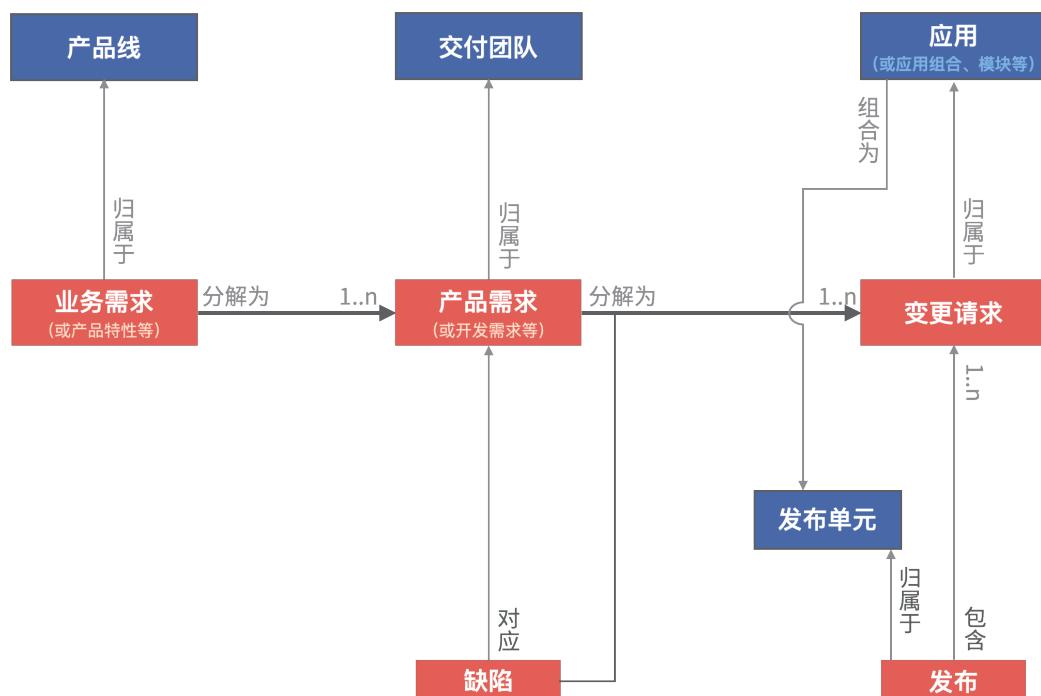


图2-2:BizDevOps概念模型的核心——作业对象和作业空间

作业空间	作业对象	具体作业链路	与其他作业对象的关系
产品线	业务需求	业务需求从输入、过滤、确认、分析、规划、实现到验收、发布、反馈、调整的整个过程	分解为1到多个产品需求
交付团队	产品功能需求	产品功能需求从创建、设计、排期，到开发、集成、测试、交付的整个过程	可能来自业务需求或产品特性；进一步分解出1到多个变更
	缺陷	缺陷的从发现、提交，到修复、验证	可能与产品需求或业务需求对应 会分解并对应变更
产品、应用	变更请求	变更从创建，到设计、编码、测试、部署的整个过程	分解自产品需求 可能被0到多个发布包含
发布单元	发布	发布从规划到集成、验证、完成的整个过程	由多个产品或应用组合而成，包含1到多个变更

明确作业对象和作业空间，为划分模型的子领域提供了依据。各个领域本身应该是内聚的，由紧密关联的概念构成。领域之间应该尽可能解耦，减少不必要的交互。围绕作业对象和作业空间划分子领域，让每个领域都有共同的目标，保证各个子领域的高内聚、低耦合。

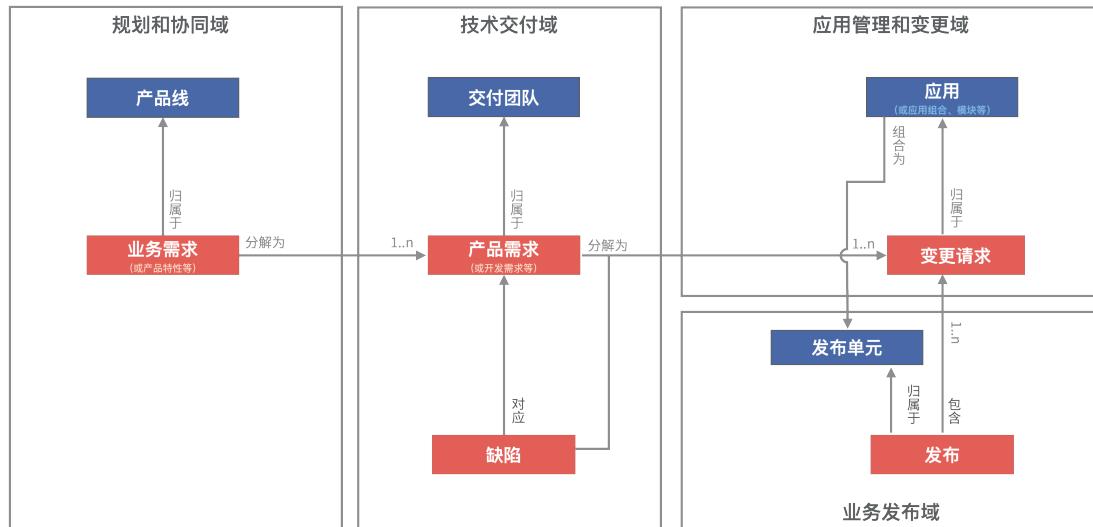


图2-3: BizDevOps模型的核心——子领域的划分

上图中灰色的框是我们对模型领域的划分，以及各个领域的参考命名。领域的划分对组织的协同、机制实践的落地、工具平台的建设都有指导意义。

• 为作业空间添加作业规则和作业计划

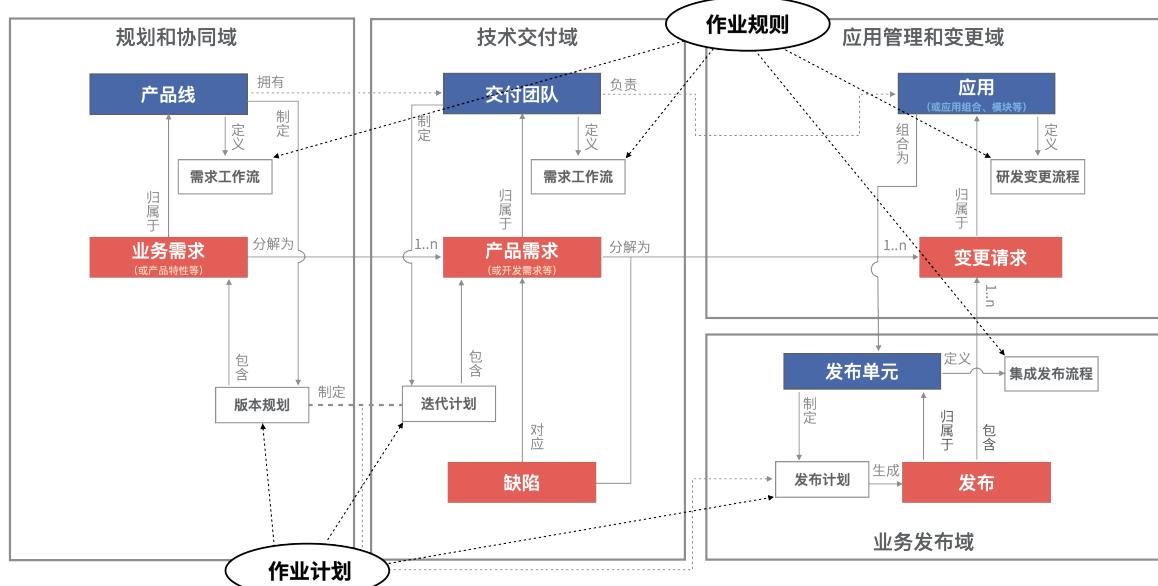


图2-4: BizDevOps模型的核心——作业规则和作业计划

划分了子领域,接下来的建模是围绕作业空间和作业对象去完善各个子领域。

如上图所示,为规范作业行为,首先各个作业空间都需要有自己的作业规则,也就是作业所要遵循的流程、规范和约束,下面是两个实例。

1.产品交付团队中的需求管理和交付流程,我们称之为需求工作流,通常包括:需求交付要经历哪些阶段,这些阶段是否可选,各个阶段的入口或出口条件等。具象到工具平台,它对应看板或需求列表中的需求状态和状态迁移规则的设定。

2.产品或应用中的研发变更流程规则,包括:变更包含的阶段,以及每个阶段由哪些步骤构成,每个阶段和步骤的入口、出口条件等。具象到工具平台中,它们通常体现为持续集成交付流水线的设计和编排。

其次是作业计划,它是在空间中对作业对象交付时间或顺序安排,如:产品线中版本规划,交付团队的迭代计划,产品发布单元的发布计划等。

• 步骤4:完成各个子领域的模型设计

模型设计的最后一步是,完成各个子领域的模型设计,典型的

还需关注的概念有:

1.作业过程需要的输入,如:产品线在规划业务需求时需要有业务的输入作为依据;

2.作业过程用到的资源,如:完成应用变更要用到的环境和流水线;

3.作业过程的最终或阶段性产出,如:发布链路会产生发布版本,并包含对应的制品,制品则是集成和交付过程的产出。

在模型中,规划和协同域里还对业务部分做了补齐,作为产品规划的输入和依据。此时,模型在其抽象层次上已经足够完整,也就是涵盖了各个价值交付链路完成作业所需要的主要概念。正如前面所述,模型完备性的金标准是它能否支持价值交付链路的高效运作。关于各个子领域的具体概念,我们将在第三部分介绍实践时展开。

下图所示是完整的BizDevOps模型的设计,除完善了各个子领域外,底部加上了基础能力域,如文档管理、代码管理、系统运维监控域等。它们中的大部分在DevOps体系中都会涉及,本白皮书不再做介绍。

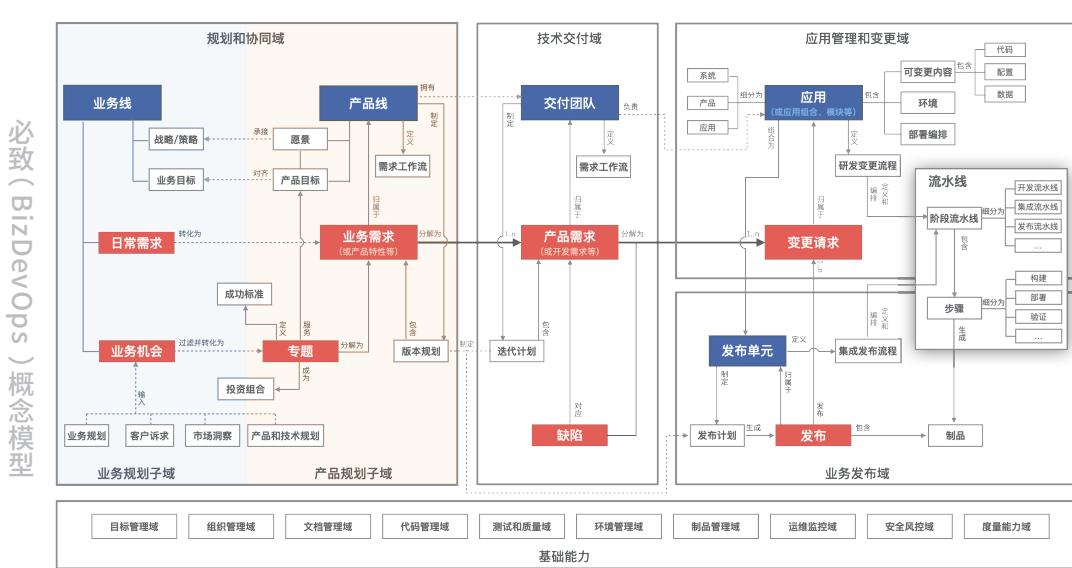


图2-5: BizDevOps整体概念模型

模型的阅读和使用说明

- 模型的表示法

上面的BizDevOps概念模型描述上使用了UML兼容的表达法。为了降低理解门槛，我们对使用的UML表示法进行了较强的约束，比如我们用“细分为”这个关联代替了UML中的以空心三角表示的继承关系，用“包含”这个关联代替了UML中的聚合和组合关系。你并不需要为了阅读这个模型，而专门学习UML。

为了让阅读者更容易抓住重点，我们用色彩标记了模型中最核心的要素——作业对象和作业空间。以下是对模型表示法的简要说明：

表示的元素	图例	内容	实例
作业对象	产品需求	价值交付链路上操作的基本价值单元，它：1) 随时间发生状态迁移和流转；2) 它的流转过程就是价值交付的过程。	业务需求，产品需求，缺陷，变更请求，发布
作业空间	产品线	容纳并操作作业对象的空间，可以在其中定义作业的规则、流程、计划，提供或链接作业所需的资源，并完成价值交付的相关作业	业务线，产品线，交付团队，应用，发布单元
其他概念	需求工作流	除作业对象和作业空间外的其他概念和实体。它们通常都服务于作业过程、或与作业过程相关，比如：作业计划、作业规则、作业过程的输入、所用到的资源、产出物等	版本规划，迭代计划，环境，配置，产品目标，愿景…
概念间的关系	分解为 → 转化为 →	表达两个概念或实体之间的关系，其中箭头表示关联的方向，文字是关系的名称。比如一个包含另一个，一个可以分解为另一个。其中实线表示两个概念间有强关联；虚线表示两个概念之间有较弱的关联。	业务需求可以分解出1..n个产品需求，应用包含自己的变更内容、配置

- 模型的使用场景

以上的概念模型定义了BizDevOps所涉及的概念、概念间的关系，并对主要领域进行了划分。它是对BizDevOps问题域相对完整的描述。它在不同的应用场景下，能起到不同的作用。

以下是我们预期的应用场景：

1. 对齐目标和策略：各类角色基于一致的模型理解BizDevOps，并对齐实施目标和策略步骤

2.为沟通建立了统一语言:帮助组织形成共同语言,保证对同样的概念有统一的理解,提升沟通的效率和效果,制定有效和可落地的行动计划

3.指导实践落地:基于统一的模型,检查当前实践中缺失或薄弱的点,特别是价值交付链路不畅和脱节的地方,落地并演进协作和工程实践,提升市场响应力

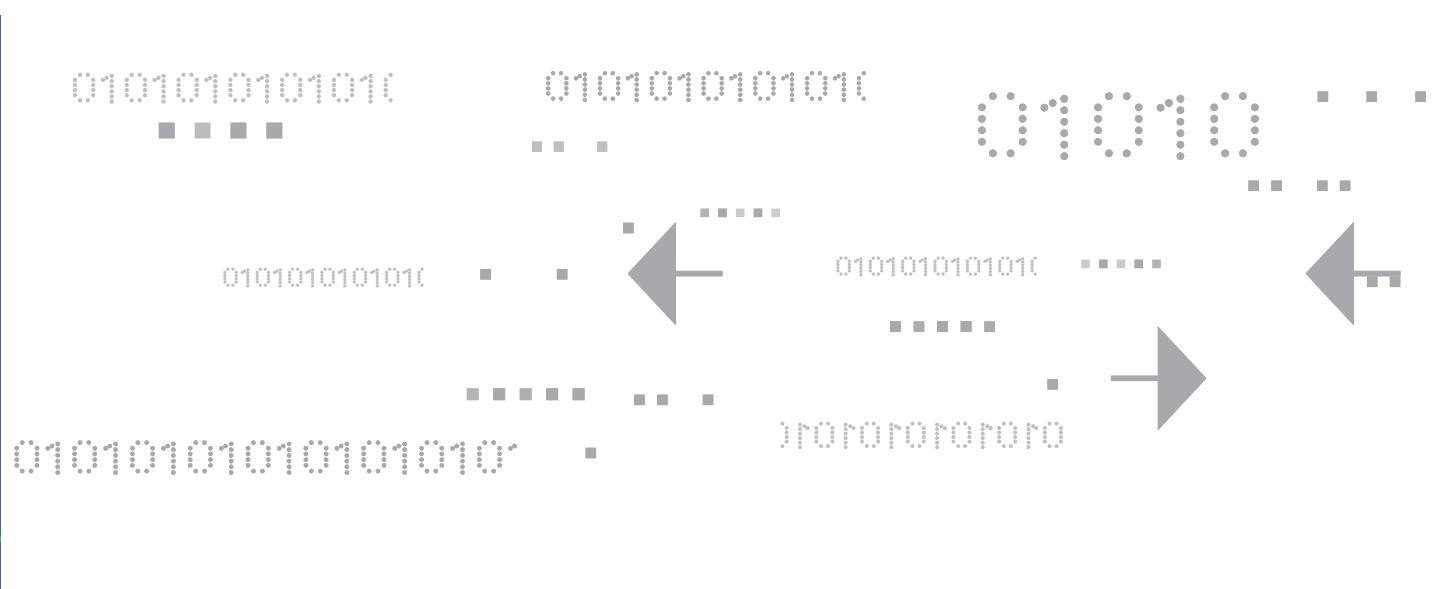
4.指导工具平台建设:指导工具平台的建设和演进,保证工具间有效的集成或一站式平台建设规划,并指导落地和推广,以及平台建设的持续性

5.指导数据体系的建设:以坚实的模型为基础,构建全量、全要素和实时的数据体系,并基于它透明交付过程,保障数字体系的有效运作,以及度量组织效能,指导持续的效能改进

6.指导数字资产的沉淀和维护:基于完整的模型,识别组织的核心IT数字资产,并持续沉淀和维护这些资产,如流程、数据、配置等资产

以上应用场景分别对应3.1节中BizDevOps落地的各项挑战。本模型希望通过清晰定义BizDevOps的问题域,来应对这些挑战,改善行业中BizDevOps的落地效率和效果。

需要说明的是,本模型的定位是参考模型,在具体的上下文中需要被定制、裁剪、细化和扩充。比如:需求类型的名称、需求的层次在不同组织会不一样;持续交付平台建设时,对工程部分的概念还需要细化。



必致 (BizDevOps) 实践体系

■ BizDevOps实践的总体介绍

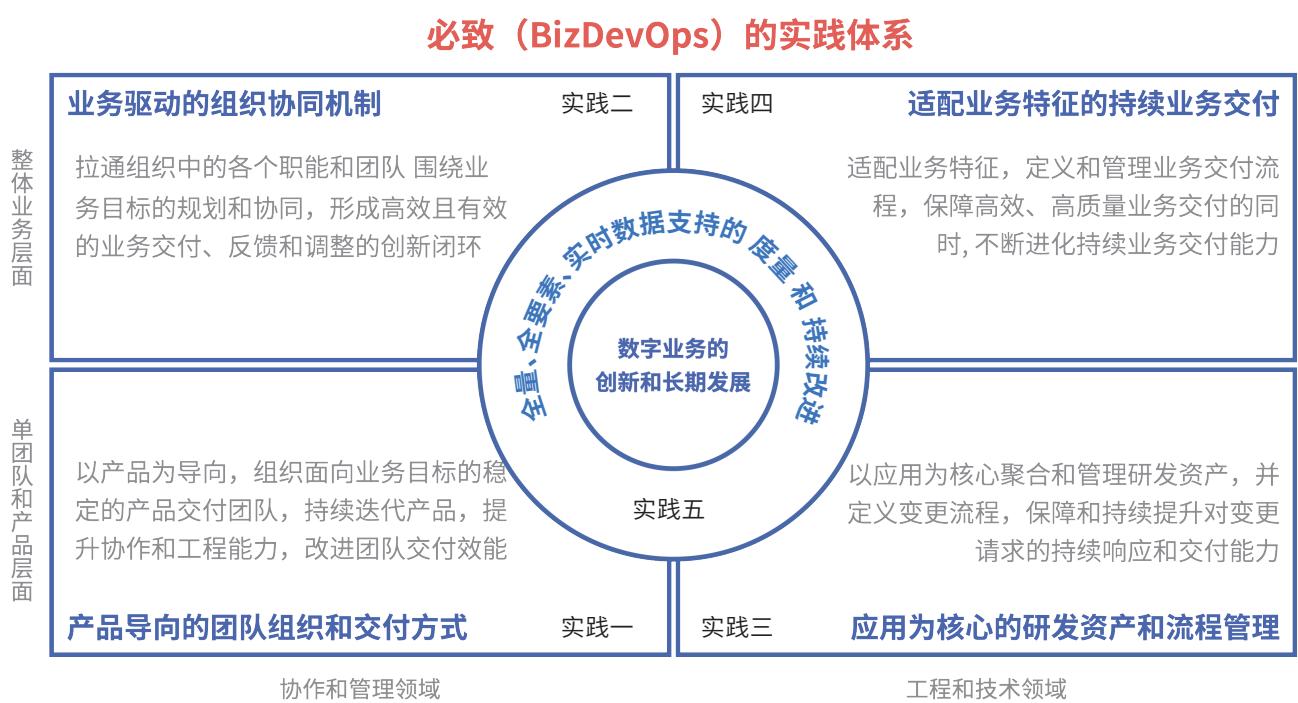


图3-1: BizDevOps实践体系由5个实践共同构成

本章将结合概念模型介绍BizDevOps的5个关键实践。我们将这5个实践分成三个领域分别介绍，也就是上图左半部分的协作和管理领域，右半部分的工程和技术领域，以及中间的度量和改进领域。

• 协作和管理领域

该领域关注从业务、产品规划，到技术交付，再到业务运营的完整链路和反馈闭环，目标是保证整个链路协同和交付的效率、质量和有效性。协作和管理领域包含两个实践。

第一个实践是**产品导向的团队组织和交付方式**，它关注单个交付团队层面，解决的问题是：如何组织交付团队，高效交付需求的同时，持续迭代产品、改进自身能力，提升交付效能？

第二个实践是**业务驱动的组织协同机制**，它关注整体业务层面，解决的问题是：如何让整个组织围绕业务目标有效协同，敏捷且合理地规划产品和响应业务，并形成有效的业务反馈、调整闭环？

• 工程和技术领域

该领域关注从技术实现,到应用变更,再到业务持续交付的完整链路和闭环,对整个链路的工程效率、质量和敏捷性负责。技术和工程领域包含两个实践。

第一个实践是“**应用为核心的研发资产和流程管理**”,它关注单个应用(或多个应用组合成的变更单元)层面,解决的问题是:如何组织研发资产和研发活动,并有效的管理和演进它们,持续提高工程响应和交付能力?

第二个实践是**适配业务特征的持续业务交付**,它关注整体业务交付层面,解决的问题是:如何适配场景落地工程交付流程,建设与业务特征相匹配的业务交付方式,打造组织持续业务交付的能力?

• 度量和改进领域

该领域包含一个实践——**全量全要素和实时数据支持的度量和持续改进**。它的目标是规划并在数字化运作中产生全量、全要素和实时的数据,并基于它们设计有效的度量用以:

- 1) 透明数字化运作过程,保障运作的质量和效率;
- 2) 衡量组织的效能、行为和能力,并建立它们的因果关联,激发持续的效能改进行动。

接下来将分别介绍这3个领域的实践。



协作和管理领域的实践

BizDevOps在管理和协作实践上关注业务、产品和技术持续地目标对齐和快速地运营反馈闭环。

如下图所示,BizDevOps的协作和管理实践是由三个闭环构成有机系统。

这三个闭环分别是:

业务交付和反馈闭环

它的核心关注点是业务的快速响应和交付,以及业务目标的达成。这一层面的核心作业对象是业务需求,并以业务目标为牵引,规划业务并统领各个产品交付团队的协同。我们用业务驱动的协同机制,来总结与之对应的实践。

产品交付和反馈闭环

它的核心关注点是产品需求的快速交付、产品的演进,以及产品交付效能的持续提升。这个层面的核心作业对象是产品需求,并以持续高效的产品交付为目标,协同各个应用的技术开发和工程部署。我们用产品导向的团队协作和交付方式,来总结与之对应的实践。

技术开发和应用变更的闭环

3.它的核心关注点是技术及工程能力的持续改进,从而保障应用为核心的持续部署和工程交付能力。这个层面的核心作业对象是应用的变更请求。它的具体实践,体现在工程和技术部分。但在协作与管理实践部分,需要被关注和连接。

基于上面的分层结构,以下从**产品导向的团队组织和交付方式**,及**业务驱动的组织协同机制**两大实践体系来阐述BizDevOps实践落地在管理和协作领域的关键。应用层面的协作更多融入到了工程和技术领域实践中。

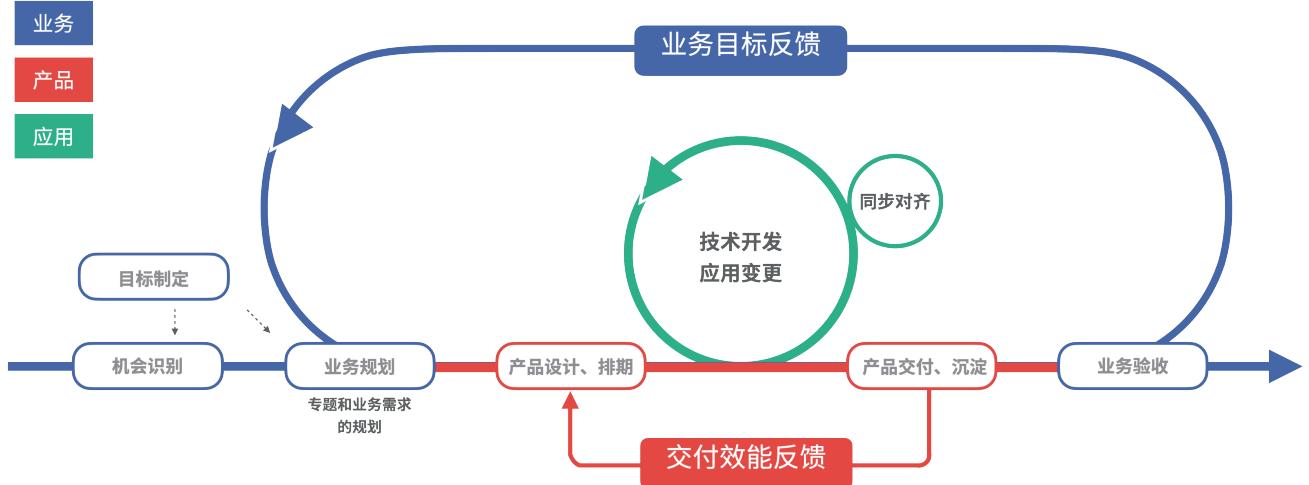


图3-2: BizDevOps关注业务、产品和应用开发三个交付和反馈闭环, 构建完整的协作实践体系

• 产品导向的团队组织和交付方式

正如本白皮书第一部分描述, 信息技术的发展进化历程, 也是技术和业务相互靠近直至融合的历程, 技术的交付模式也随之进化。

在最初技术与业务相对分离的模式下, 业务确定要什么, IT就开发什么。而需求确定后, 则变成IT开发了什么, 业务就用什么。IT通常被当做成本中心看待——花费确定的预算交付确定的内容。

在这一模式下, 大部分IT交付是以项目的形式进行的。项目的内核是: 在特定时间内, 以相对确定的预算和人力, 交付预先规定的内容。项目追求确定性, 注重计划和计划执行。在技术与业务分离的时代, 这样划清边界, 对双方都有好处。相应的, 组织也可以按项目来预先规划、分配和执行IT预算, 而这通常是以年度或至少以季度为单位的。

项目在组织方式上表现为, 团队的短期性和临时性。人被作为资源分配到确定好内容的项目中, 而项目结束后则被回收, 准备分配到下一个项目。这样的组织方式不利于技术和工程资产的长期积累和优化, 也不利于工程能力及基础设施的持续优化。

数字化时代, 技术成为业务发展和创新的核心动力, 加之技术和业务的不确定性都越来越高, 技术与业务分离的方式越来越不能满足即时响应业务的需要, 也无法做到紧密协同下的持续迭代, 以及工程及技术资产的持续积累和改进。

适应时代变化, 在业技协同方面的关键认知转变就是从项目到产品的思维过渡。面向BizDevOps的业务、产品和技术的协同, 要求不同职能单元转换视角, 形成产品导向的交付模式, 面向数字资产的长期演进和持续沉淀而通力合作。我们将项目和产品导向交付模式的关键差异对比如下表。

	项目导向的交付模式	产品导向的交付模式
基本假设	假设项目范围内的未来是确定的，寻求按计划交付预先确定内容。	拥抱不确定性，建立反馈闭环持续探索并发现价值。
成功标准定义	从把产品技术看作成本中心，以按时和按预算交付来衡量成功。	到把产品技术看作利润中心，以业务目标和结果的达成来衡量成功。
工作分配模式	从把人作为资源分配到预先确定的工作（项目范围）上，倾向于根据专业分工批量式的规划和交付。	把工作（业务需求）分配到稳定的价值交付团队，寻求持续迭代交付和演进，倾向于根据产品交付要求组建跨职能团队。
关键产出	关注项目一次性的交付结果。	关注产品的长期演进以及资产沉淀和应用。

下图是BizDevOps概念模型中的“技术交付域”及其相关联的部分，它为产品导向团队组织的交付方式提供了基本依据。对应该模型，各产品线定义并维护产品线的愿景和目标，并建设相对稳定的持续交付团队。

各个交付团队：

-  共同对产品线的愿景和目标负责
-  承接业务的需求，不断迭代产品，服务产品和业务目标
-  对交付效能负责，持续改进自己的协作方式、工程和技术能力
-  负责1到多个应用，并对它们的演进和长期质量负责

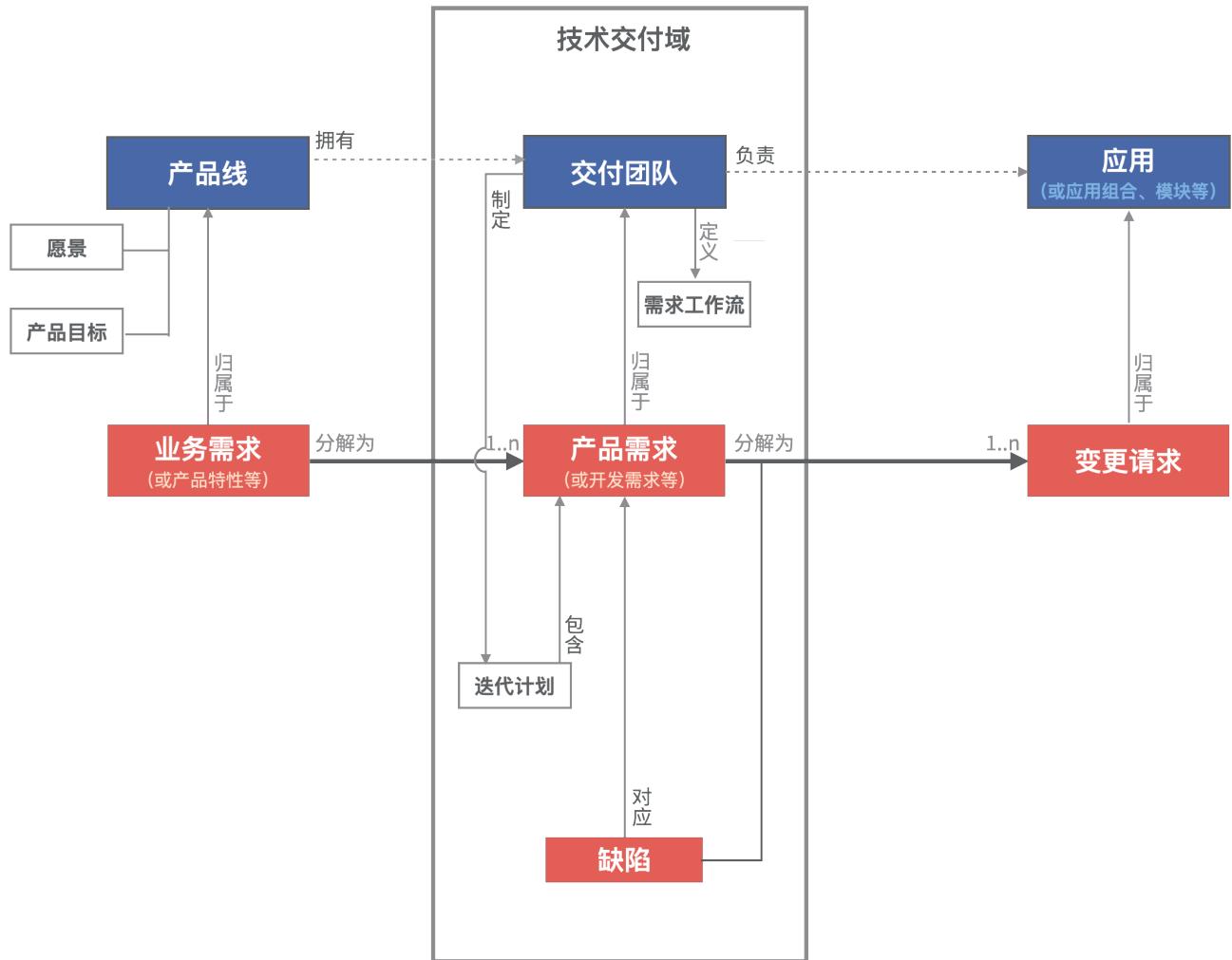


图3-3:产品导向的团队组织和交付方式定义的技术交付域

虽然交付团队的人员更迭在所难免,但团队在运作过程中应该面向长期目标,寻求可持续发展。同时,团队应关注核心角色的梯队建设,保证产品需求交付相关核心能力的延续性。长期可持续发展的交付团队也是保证产品内部和外部质量的基础。由于对产品交付长期负责,团队就有动力持续保证产品的高质量,类似持续重构这样面向易改变性的质量实践,才能够得到有效落地。

交付团队针对产品需求定义“需求工作流”,承接业务需求分解得到的相关产品需求,并针对下游相关应用,明确变更请

求。由于产品和技术上下文的不同,具体的工作方式可以授权团队自己定义,只要团队对最终的交付效能负责。

践行产品导向也要求我们拥抱,而不是害怕和忽视不确定性。交付团队需要根据不同的业务和技术上下文,选择合适的迭代频率、制定相应的“迭代计划”,根据一定的优先级策略明确迭代内需要交付的产品需求,并采用合适的敏捷和精益方法,来管理团队的交付。关于敏捷和精益方法的具体操作,已经有很多参考,不在本白皮书范围之内。



以产品为导向，组织面向业务目标的稳定的产品交付团队，持续迭代产品，提升协作和工程能力，改进团队交付效能。这是产品导向的团队组织和交付方式解决的核心问题。



• 业务驱动的组织协同机制

产品导向的团队组织和交付方式，解决的是团队层面和产品交

付的问题。然而，产品必须与业务对齐，才能交付有用的价值。

很多企业将业务价值等同于短期经营的财务指标，造成业务和产品、技术无法建立价值视角上的共同语言。此时，产品和技术团队就容易将工作成效定位到内部交付工作上，比如代码的产出、功能交付或缺陷修复数等，造成了产品与业务的价值脱节。

价值最终必须从外部客户和市场的视角定义，业务策略和目标也要源自客户和市场。另一方面数字化时代，业务策略和目标通过数字化的产品才能够实现。只有包括业务和产品技术在内的整个组织协调一致，打通价值的探索、规划和交付链条才能实现真正的高效业务交付。然而，业务和产品应该如何协同？它们之间的关系应该是怎样的呢？

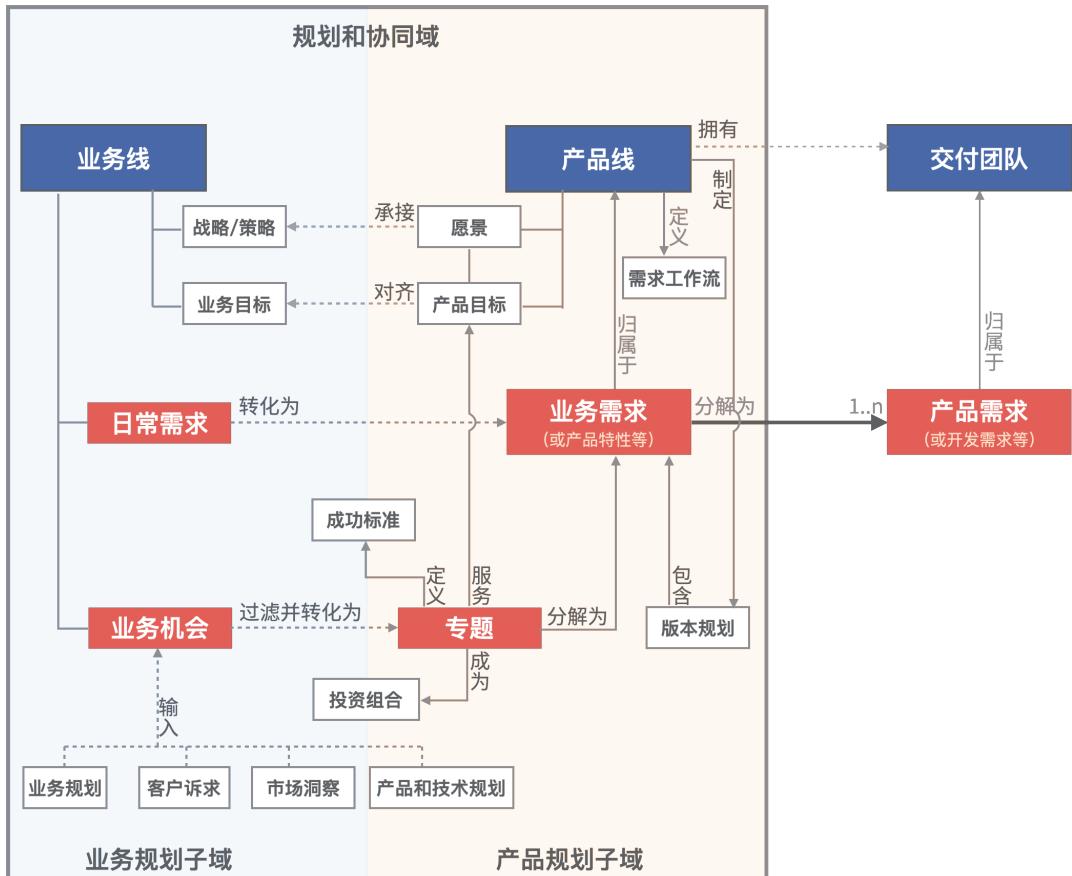


图3-4:业务驱动的组织协同机制定义的规划和协同域

如上图所示：BizDevOps概念模型述了业务与产品的关系，我们称这一关系为：“业务驱动，产品收敛”。其中，业务是规划的驱动源头。但，最终业务的输入需要在产品侧被收敛，才能落地为驱动数字业务进化的具体内容，并完成交付。“业务驱动，产品收敛”，是数字化时代，业务与产品关系的最佳范式。

在整体规划上，各业务线基于客户定位、市场环境、竞对情况等确定业务的战略或策略，并定义业务目标。而产品线也会综合业务的输入和产品技术的规划，定义产品愿景和产品目标。在现实中，一个产品可能需要支持多类业务或客户。但，产品一定要向所支持的业务对齐，也就是产品愿景承接和支持业务战略，产品目标对齐业务目标。这是在整体规划层面的业务驱动和产品收敛。

在具体的操作层面，业务对产品有两类输入。一类是支持现有业务和客户的较为琐碎的日常需求，它们会被过滤并直接转化为产品的业务需求，并按优先级计划和交付，以支持现有业务的运作。一类是更大粒度的面向未来的需求，我们称之为业务机会。业务机会需要被进一步分析，综合其预期成效、成本、紧迫性等要素，如果这个机会值得投入，则会转化为产品线下的“专题”，进行具体规划和交付。

专题指的是业务和产品线为抓住业务机会而采取的具体行动。一个业务机会可能产生一个或多个专题，多个关联的业务机会也可能合并成为一个专题。专题与过去的研究项目有类似之处，但又有本质的不同。相对项目，专题更强调拥抱不确定性，基于专题的定位和成果定义，团队动态规划专题的具体内容，并在过程中持续迭代、反馈和调整。考核专题成功与否的并非是按期交付预先确定的内容，而是达成或超越所定义的成果，以及从中获取有效的产品和业务认知。

BizDevOps概念模型中，规划协同域被分为业务规划子域和产品规划子域，这两个子域既相对独立又紧密协同，是BizDevOps所倡导的业技协同的重要组成部分。

业务规划子域

在数字化业务中，需要针对业务规划、客户诉求以及市场洞察等持续进行业务机会的发现和识别。在技术日新月异的科技时代，从产品和技术侧的洞察和机会也必须得到重视，因为数字化本质上是用技术解决业务问题，产品和技术侧的规划也是业务机会的重要输入。

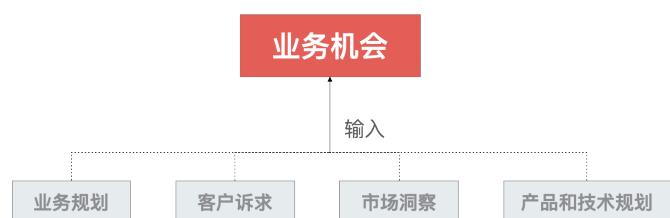


图3-5:业务机会的输入主要来自业务侧，但也包括产品侧的规划

值得注意的是，在梳理业务战略和目标时，需要对业务价值进行分类建模，这样才能驱动后续在产品和技术侧的价值对齐。常见的价值模型需要关注三个不同的视角，也就是：



以上三个视角，可以指导业务或产品发现和定义价值指标，并基于企业的原则设定业务和产品的综合目标。对高不确定性的创新业务，在特定的阶段选取单一指标用于凝聚整个产品和业务团队的方向是常见的做法，这一单一指标被称为北极星指标，它可以帮助团队聚焦核心价值点，并推动更快速和频繁的价值验证闭环。随业务阶段的推进，北极星指标也要随之调整。

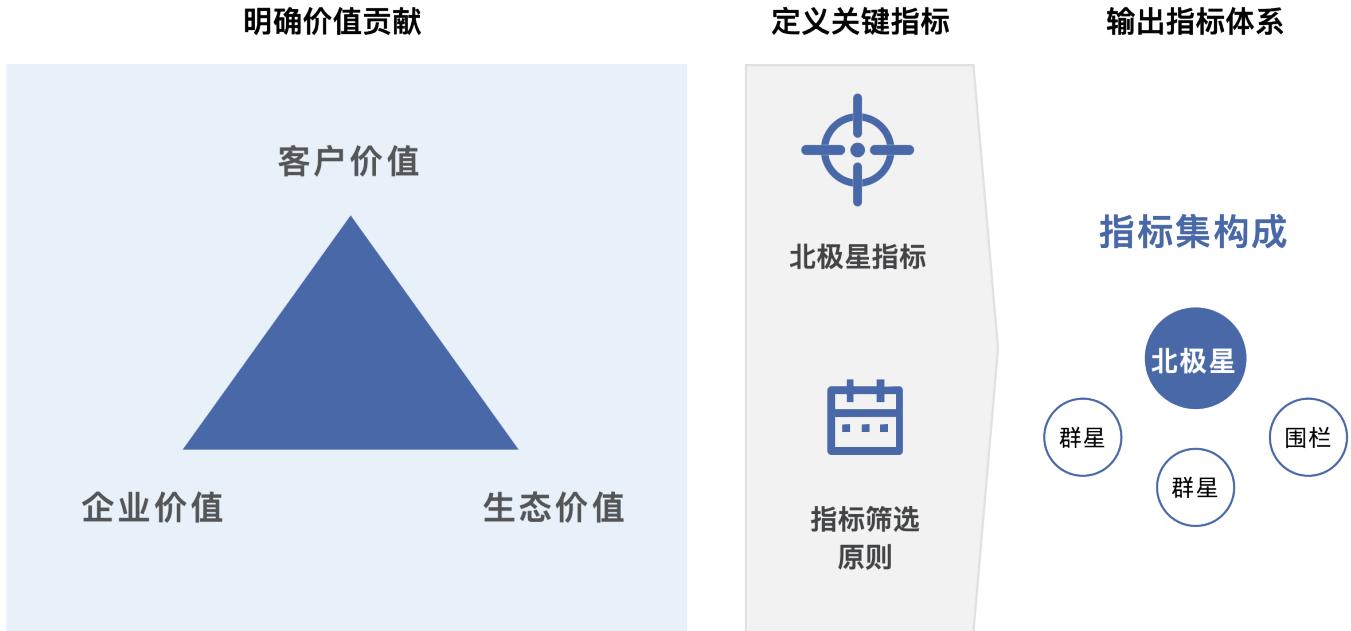


图3-6:从三个不同价值视角出发建立价值指标体系模型,融合单一北极星指标模式来明确阶段性重点

产品规划和协同子域

在产品规划和协同子域中,业务机会通过专题的收敛和优先级排序分解为相关的业务需求,同时日常需求也是业务需求的来源。面对数字业务的蓬勃发展,业务机会和日常需求总是源源不断,但产品和技术团队的资源有限,常常造成企业内部业务线和产品线矛盾不断,在业务需求优先级上很难达成共识。

BizDevOps强调业务驱动的协作关系,由此也需要围绕业务需求建立共识机制。业务机会通过专题进行收敛和定义,专题服务于产品的目标,并定义明确的成功标准,它又被称为成效指标(MoS:Measure of Success)。成功标准应该明确可衡量,它由业务和产品共同定义并达成共识,用以指导专题的规划、优先级共识和迭代反馈。成效指标的衡量则以业务的输入为主,为业务驱动、产品收敛画上闭环。

成效指标(MoS):专题的成效指标是专题在启动时确定的

如何衡量其结果的标准。在具体落地时需要关注数据的可获取性,逐步形成自动化的获取、处理和呈现机制。

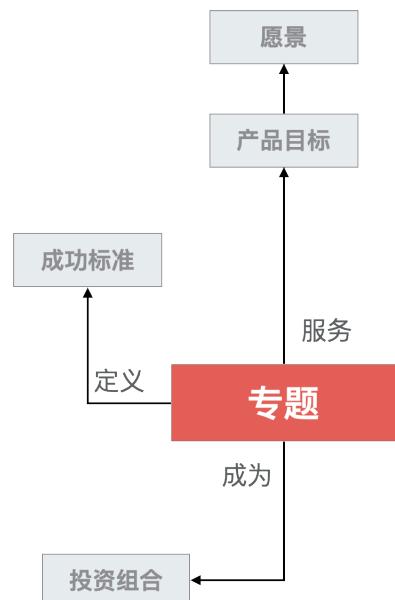


图3-7:通过专题收敛业务的机会,并承接和落地产品的目标

成效指标不应该等同于业务经营的KPI,整个指标体系为业务、产品和技术在协同过程中提供了仪表盘,保证大家在高速前行过程中保持方向感。过度聚焦短期的KPI,则容易迫使组织中各团队急于证明自身工作的高效,形成不利于响应变化的免责文化。很多传统组织在面对数字化时代不确定性时,尝试引入OKR等机制去对抗长期KPI管理的副作用,然而在目标(Objective)的制定上,仍然很难聚焦真正的业务价值,造成落地执行的乏力。

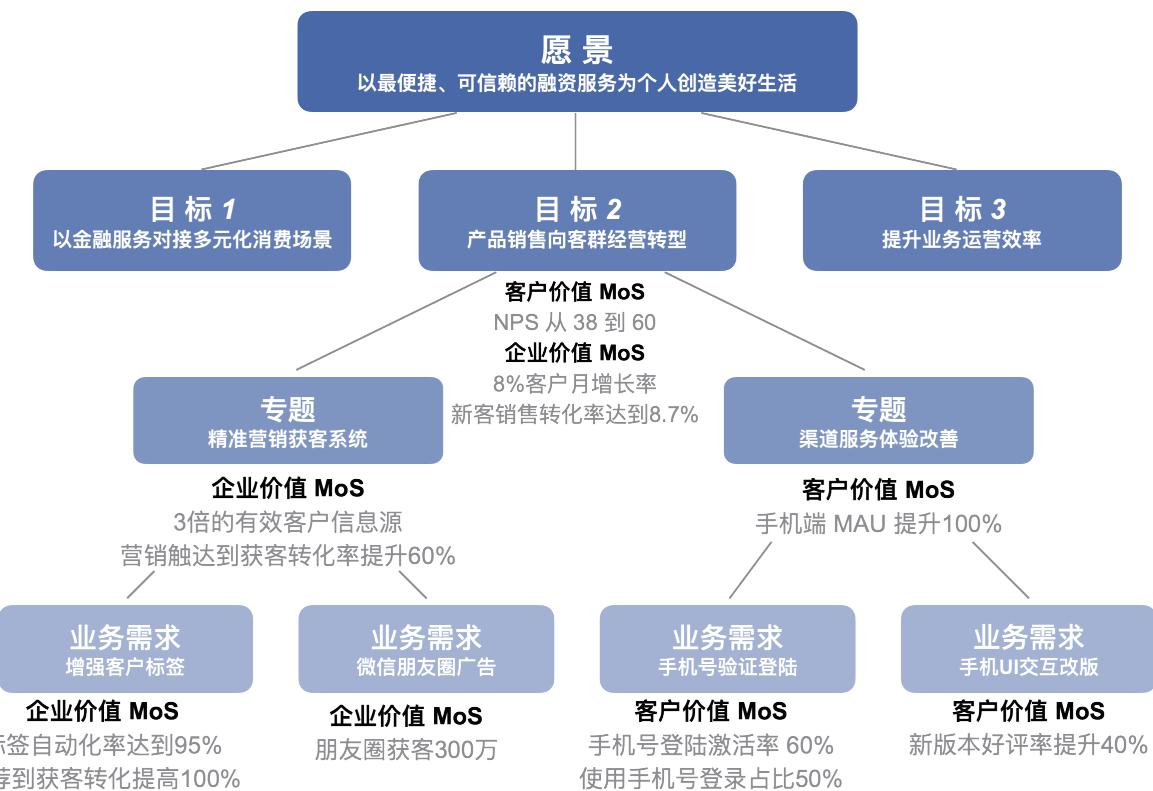


图3-8:利用精益价值树展现的从业务愿景/目标,到专题和业务需求的投资组合管理。

如上图所示,产品线的愿景、目标、专题和业务需求构成树形结构,树的各个层次反映不同粒度的业务价值。这棵树被称为精益价值树(LVT:Lean Value Tree),精益价值树的不同分支上叠加人力和其他资源的投入,可以整体和直观反映组织中的资源在不同价值项上的分布。

精益价值树可以帮助组织进行基于业务价值的投资组合管理,通过可视化的实践让业务和产品在规划和协同上持续互动,保证业务和产品在决策上的透明,和沟通渠道的有效畅通。



围绕业务目标的规划和协同,形成高效且有效的业务交付、反馈和调整的创新闭环。这是业务驱动的组织协同机制解决的核心问题。



在基于专题的投资组合管理下,组织进一步针对业务需求形成有效的版本规划,并以它进一步驱动交付团队的迭代计划,最终转化为以及持续交付域的发布计划,拉通协作和管理领域与工程和技术领域,形成完整的BizDevOps体系。接下来一节我们将介绍工程和技术领域的实践。

工程和技术领域的实践

BizDevOps在工程和技术领域的核心目标是:建设组织的技术工程体系,保障并持续提升业务的持续响应和交付能力。

为此需要关注:



产品的持续开发和部署能力,它是工程能力的基础

业务的持续、高质量发布能力,它是对业务的价值的最终体现

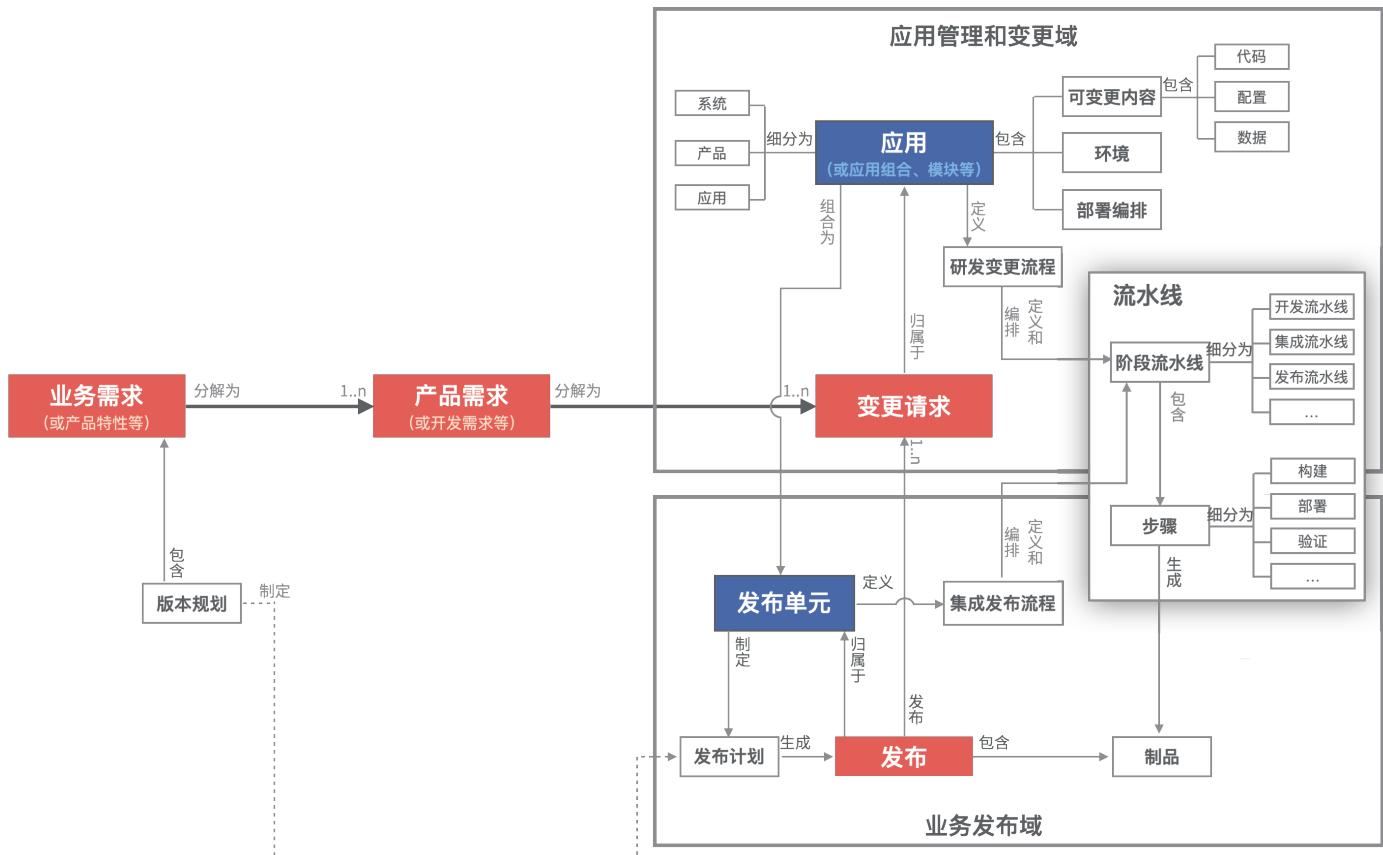


图3-9: BizDevOps模型的技术和工程部分

在BizDevOps概念模型中,我们把工程和技术域分成两个2个核心域——应用管理和变更域,业务发布域。

1.应用管理和变更域

变更请求是这个域的核心作业对象,它指的是为实现特定的产品需求或修复缺陷,应用所要做的变更。变更请求的实现是从设计、编码、提交、测试直至部署的完整过程。变更请求和应用这两个概念,后续会给出完整定义。

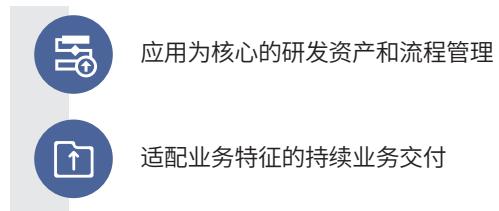
应用管理和变更域的目标是:以应用为核心,有效聚合和管理研发资产,定义和管理应用的变更流程,保障对变更请求的持续响应和持续部署。

2.业务发布域

(系统或版本的)发布是这个域的核心作业对象。一个发布通常是基于发布计划生成,并按发布单元所定义的集成发布流程,完成多个变更请求的集成、验证和对外发布。关于发布和发布单元,后续会给出完整定义。

业务发布域的目标是:适配业务特征定义和管理业务的集成发布流程,制定发布计划,保障高效、高质量和持续的业务响应和交付。

以上两个域,分别对应两个关键实践:



接下来的两节将分别介绍这两个实践。

- **应用管理**

在数字化进程中,保障和持续提升业务交付能力,是企业核心挑战之一。交付能力的提升是一个持续的过程,需要全链路上各个角色的高效协同。而在云时代,“应用(Application)”是工程上打通各个角色协作的关键,开发、测试和运维的工作都应该围绕应用展开。

应用是云原生环境下最基本的部署单元和运维对象,它由1到多个组件构成,运行在特定的基础设施之上,对外部提供服务或功能。应用拥有自己的代码、配置和数据,可以被单独变更、部署。应用之间应该是松耦合的。

关于应用的管理,本白皮书承接了**OAM** (Open Application Model, 开放应用模型)模型标准。OAM模型认为应用是开发者、应用运维、基础设施运维的共同的工作对象,不同环节基于应用拉通并有效协作,成为一个高效工作的整体。

Open Application Model

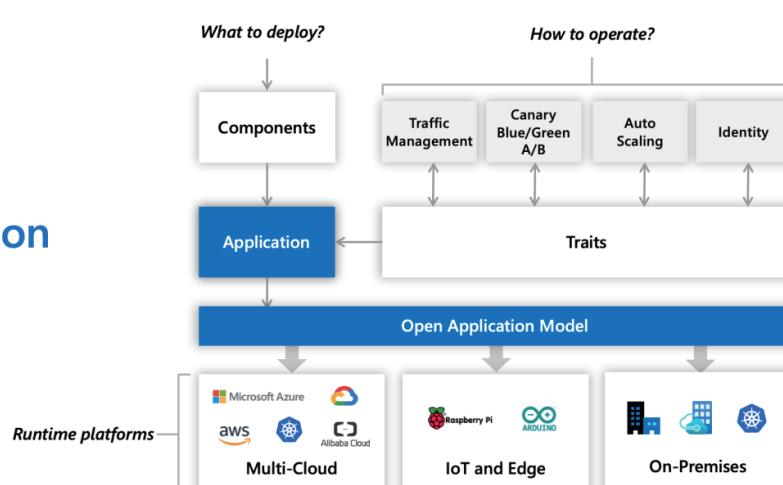


图3-10:OAM模型* (来源<https://github.com/oam-dev/>) 定义了应用的声明方式,并以应用打通部署、应用运维和基础设施运维

如上图所示, OAM模型中应用的声明定义了应用的部署组件、发布策略、运维策略等。BizDevOps将应用管理的范围延伸到研发阶段, 这样在OAM打通部署、应用运维和基础设施运维的基础之上, 进一步向前打通应用的开发环节。为此, 应用应该:



图3-11: 变更请求是应用管理和变更域的核心作业对象

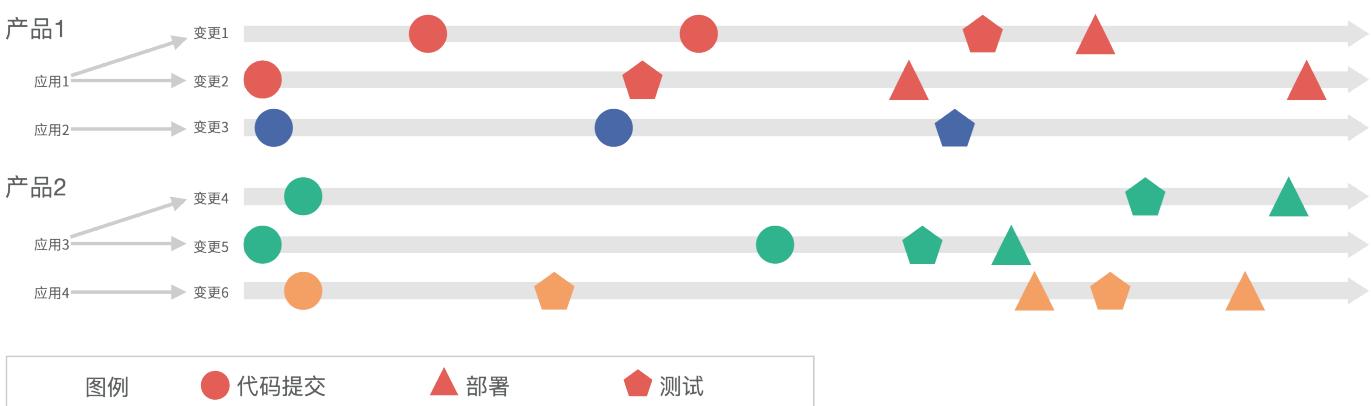
应用响应来自外部的工作请求, 我们称其为变更请求。上图表表达了变更请求在工程活动中的位置。对外, 变更请求承接了产品需求针对某个应用的所有需要进行的技术活动; 对内, 变更

请求归属于特定的应用, 遵循应用所定义的研发变更流程, 并以统一的流程聚合开发、集成、测试、部署等技术活动。

变更请求:为实现特定的产品需求或修复产品中的缺陷, 某个应用所需要完成的工作。变更请求的实现, 是包含设计、编码、提交和部署等在内的完整过程。与分散的技术任务不同, 变更聚合了为响应特定产品需求, 应用上所发生的所有技术活动。

如下图所示, 在BizDevOps模型中, 我们将创建变更请求作为工程和技术活动的起点, 而变更请求则遵循应用所定义的变更流程, 聚合包括设计、编码、测试、部署等在内的研发活动。这一机制打通了从需求协作到工程部署的链路, 持续改进变更流程的效率和质量是工程效能的基本保障, 也是高效业务发布的基础。

以应用为核心聚合和管理研发资产, 定义、管理并持续改进应用的变更流程, 实现对(来自业务和产品的)变更请求的持续响应和持续部署, 它是BizDevOps中工程交付效能的基本保障。



- 适配业务特征的持续业务交付

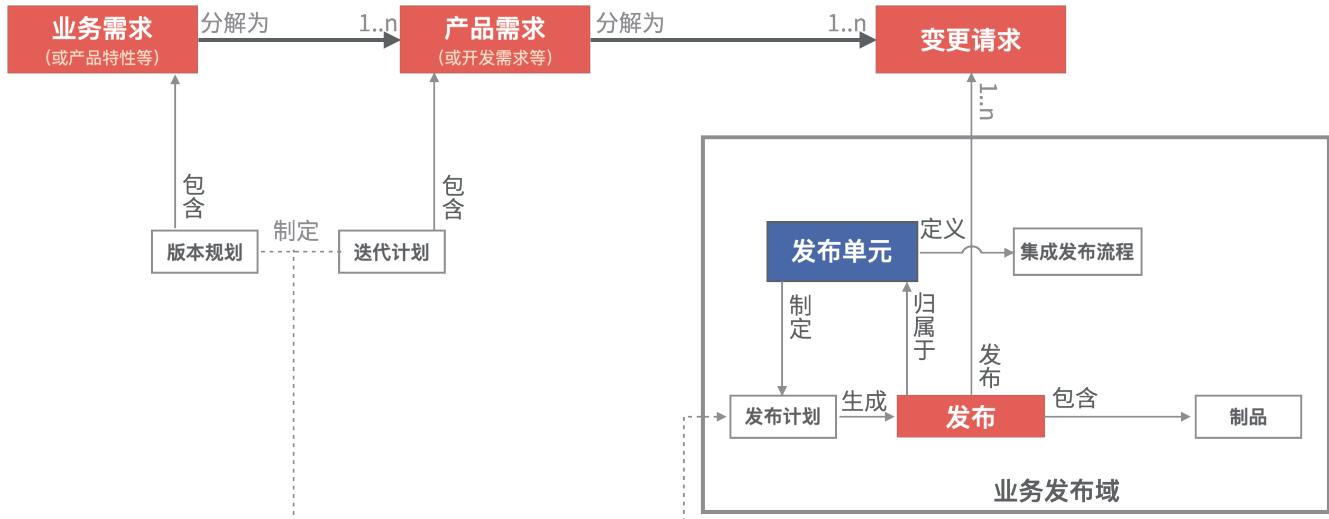


图3-12:BizDevOps概念模型中关于业务发布和交付的部分

BizDevOps实践的最终目的是提升业务交付能力。在工程和技术上,业务交付体现为版本或业务的持续发布。

发布是指:按照一定的流程,将某个发布单元(如产品和系统)中开发完毕的功能,分发到目标环境中,并使之对外部可见和可用的过程。

发布具备如下特征:

- 发布由发布计划生成,发布计划通常源自业务的版本规划,它规定发布的业务内容,如:包含哪些业务或产品需求,缺陷修复等
- 发布是针对某个发布单元的。发布单元可以是单个应用,也可以是一组应用的组合而成的产品、系统等

- 每次发布通常会发布一系列开发完毕的变更请求,这些变更请求都属于发布单元所涉及的应用
- 发布要遵循发布单元定义的集成发布流程。发布流程通常由流水线承载,如:集成阶段流水线、发布阶段流水线等,这些流水线可以被编排而成为整体的发布流程,并生成发布的制品

发布:指将开发完成的功能,分发到目标环境,并使之对外可见和可用的过程。

发布单元:是发布的最小粒度,它可能是单个应用,或有多个应用组成的产品和系统。发布单元下的变更请求需要一起集成、验证、部署后才能共同发布,发布单元定义自己的集成发布流程,以保障发布的可控性和质量。

兼容“稳态”和“敏态”的BizDevOps工程交付标准模型

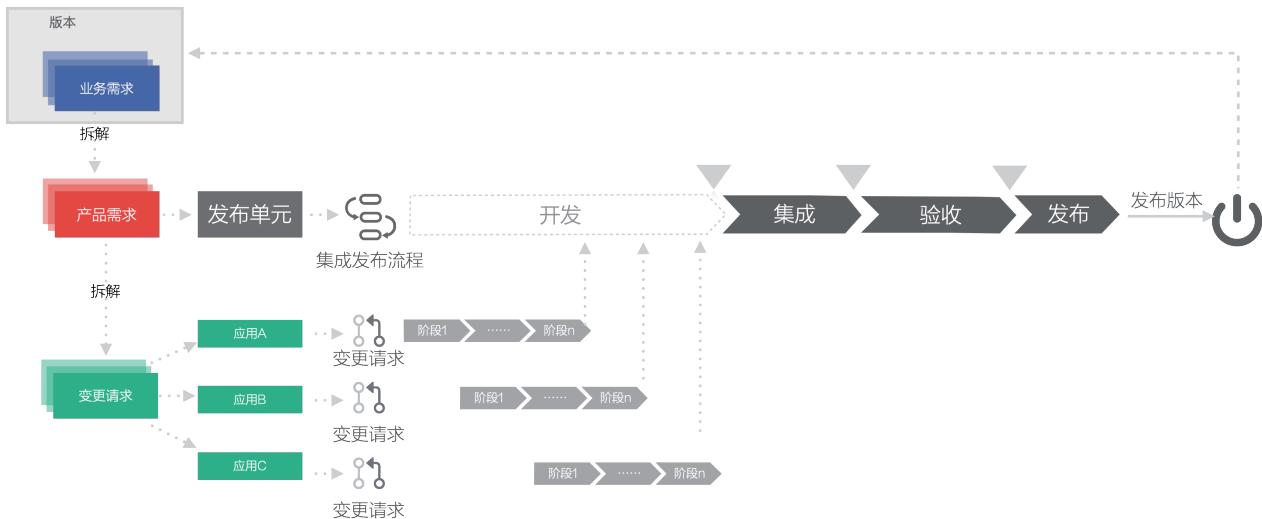


图3-13:业务交付的标准过程模型

上图是业务交付的标准过程模型，描述了业务交付中涉及的概念和流程。其中，一个业务版本包含1到多个业务需求，每个业务需求拆解为1到多个产品需求；产品需求又被拆解为1到多个应用的变更请求；变更请求按所在应用的变更流程开发完毕后，成为发布流程的输入。

发布则是以发布单元为单位的，一个发布单元包含1到多个应用，这些应用中的变更请求必须在一起集成和验证后才能够发布，并生成统一的发布版本。发布需要遵循发布单元所定义

的集成发布流程，流程中的步骤可能是自动触发和运行的，如：流水线驱动的自动测试或部署；也可能是手工的，如人工的测试和线下的审批。

上面的标准过程模型，并不刻意区分交付是传统的批量交付或敏捷的持续交付，也就是在一些标准中所声称的稳态交付模式和敏态交付模式。我们认为，敏态和稳态并不是泾渭分明的两种方式，而是一个连续的过程，任何业务都有变得更敏捷的权利和需求。

稳态或敏态的交付和发布是一个连续的过程



图3-14:稳态和敏态的交付和发布是一个连续的渐变过程

基于上面的标准过程模型,发布模式更偏向批量、强管控的稳态发布,还是灵活和持续敏态发布,主要依赖于下面三个要素:

要素一:发布单元的粒度大小

在稳态模式下,发布单元的粒度通常较大,比如多个应用甚至多个子系统构成发布单元。一个发布单元下的所有变更请求需要联合发布。敏捷的持续发布模式,寻求减小发布单元的粒度,最极致的敏捷状态是单个应用可以作为发布单元单独发布。

减小发布单元的粒度也是有前提的,它需要各个发布单元之间的解耦,包括技术上的解耦,各个发布单元可以独立验证,但依然可以保障系统的质量。实现这一点需要持续的工程和技术能力的改进。

要素二:发布的批量大小

在稳态模式下,通常会以大版本的形式成批量的集成、验证和发布,每个版本发布的间隔从数周到数月不等,发布的内容则包含这段时间内积累的所有待发布的请求。敏捷的持续发布模式寻求减小需求发布的批量,最极致的敏捷状态是单需求发布。

减小发布批量并非全无代价。当单次发布的成本较高时,减小发布的批量会带来额外的成本。同时,研发的模式也会制约发布批量的减小,比如:在瀑布研发模式下,需求批量开始和批量完成,批量地发布也就成了自然选择。

要素三:集成发布流程的灵活性和速度高低

集成发布流程指的是发布单元发布前所必须遵循的流程,如:对变更请求的集成、在测试环境上的部署和验证、业务的验收、生产环境的部署、灰度或全量发布,以及相关的检查和审批等。

在相对稳态的模式下,集成发布流程通常比较复杂,且耗时长。比如,繁杂的集成工作,过度依赖集成发布阶段的测试来保障质量,大量长耗时的手工工作,各个节点需要严格的人工审批,以及过多的返工和问题排查等。

敏捷的持续交付模式,则希望尽量降低集成发布流程的复杂度,缩短发布的耗时。最理想状态下,发布工作能够自动完成,耗时极短(比如在数分钟内完成),并能够保障发布的质量,即使出现问题也能够及时发现和自动回滚。

提高发布的灵活性和效率也是有前提的,它要求我们不断进行质量保障前移,提高待集成制品的质量,减少发布过程中所需要的验证工作,并尽量让发布过程自动化,不断提高这一过程的质量、效率和可观测性。

以上三个要素,在保障交付质量的前提下,共同决定组织的持续交付水平,我们称之为“**持续交付的三要素**”。持续交付的目标非常直接,那就是持续地交付业务价值。其中,业务价值的载体是业务需求,而持续指的是在需要的时候随时可以交付,而非受制于批量要求、复杂的发布流程、落后工程能力等。

BizDevOps 工程交付理想模型
单应用部署、单业务需求发布、最小化发布流程 是 持续业务交付 “完美” 目标

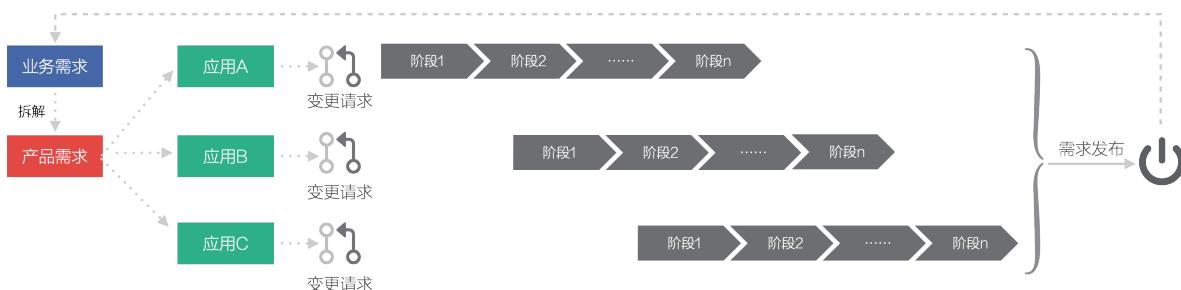


图3-15:按应用部署,按需求发布,发布流程最小化是持续交付的理想模型

上图是持续交付的理想模型，它与前文的标准模型是兼容的，是标准模型下的一个特殊版本。它可以表述为：



按单应用持续部署

发布单元是单个应用



按单业务需求持续发布

业务发布的批量是单业务需求



最小化的发布流程

将集成和质量保障等技术活动，尽可能左移至单个应用的变更阶段；将集成发布流程尽量限定为业务意义上的发布——以业务需求为单位，让已部署的功能对外可见。这也是当前技术条件下除无服务器架构外最极致的持续发布状态了。

达到理想状态固然不易，但在实现上是可行的，卓越级的互联网产品一天能完成几十到几百次发布，通常就是以达到这样的状态为支撑的。然而对于任何组织，持续交付能力都受制于两个方面的因素。

第一个制约因素是业务的特征

例如：客户侧部署的软件产品，就很难做到在生产系统上的持续部署。尽管远程升级、无感知发布等技术降低了发布成本和风险，但与互联网产品相比，持续发布要困难许多。此时，首先应该追求内部版本的持续交付，持续获得质量反馈，并降低客户侧发布的风险，并在此基础上不断降低客户侧交付的成本，为持续交付创造条件。

第二个制约是团队技术和工程能力

例如：如果团队自动化水平低，单次集成和验证的成本过高，此时一味的降低发布的批量是不现实的；技术架构上不能解耦，发布单元间的解耦就无法实现，频繁发布或带来负面影响；应用级别的质量不能保障，又缺乏全链路的质量验证能力，一味简化集成发布流程，反而可能适得其反，损害质量和效率。

综上所述，持续交付能力体现为高质量持续交付业务价值的能力，持续交付能力的建设是一个渐进的过程。组织的交付模式需要与业务特征相匹配。在此基础上，组织应该不断提升技术和工程能力，在保障质量的前提下，向持续交付的目标迈进。交付模式应该随着技术和工程能力的提高而不断进化，上面的持续交付的理想模型则为进化提供了方向。



适配业务特征，定义和管理业务交付流程，保障高效、高质量业务交付的同时，不断进化持续业务交付能力。这是组织打造业务持续交付能力的目标和努力方向。



度量和改进实践

业务是BizDevOps的起点，是贯穿价值交付链路的核心要素，也是衡量BizDevOps实施成功的终极标准。同样，BizDevOps的度量体系，也必须以业务为核心，服务业务的成功。

• BizDevOps度量的目标和指标体系

BizDevOps度量的目标是：指导团队针对性地改善能力和行为，从而提高交付效能，最终带来期望的业务结果。基于这一目标，我们把度量指标分为三类。如下图所示，它们分别是：业务结果指标、交付效能指标，以及能力和行为指标。区分这三类指标并明确它们的关系是有效度量的基础。

业务结果指标：右边的业务结果指标反映业务的效果，如：收入、利润、用户数、转化率、留存率、服务成本、用户口碑、社会价



图3-16: BizDevOps的本质目的和指标分类

值等。它们是组织的最终目标，但对产品研发而言，这些是既成事实的结果，无法用来直接指导改进行为。

交付效能指标：中间的交付效能指标用以衡量产品研发的对外（业务）交付效能，如：响应周期、吞吐率、质量和有效性等。交付效能指标是产品研发侧关于业务结果的代理，我们也称其为“代理指标”。作为代理指标，它与被代理的业务结果之间的转化函数是有损的。我们应该选取并关注那些对业务结果转化效能高的指标，把它们作为核心的效能指标，并持续检验它们与业务结果的关联度。尽管交付效能指标反映了研发的对外交付效能，但还是不能用来直接改进。

行为和能力指标：左侧的行为和能力指标，包括团队和个人的行为，如：工程习惯、协作方式、需求并行度、代码评审质量等；也包括技术和工程能力，如：代码质量、架构耦合、测试覆盖率、发布成功率等。它们与交付效能之间转化函数同样是有损的。尽管，理论上行为和能力决定交付的效能；但，我们需要系统分析哪些能力和行为影响了交付效能，才能针对性地改善能力和行为，从而提高交付效能，带来期望的业务结果。

上面的三类指标中，越靠右边的是越外部的，接近想要的结果；越靠左边的是越内部的，指向需要改进的行动。效能度量中常见的误区是：把内部指标作为衡量和考核要求。它带来的问题是：内部数据提升了，交付效能和业务结果却没有改变，甚至因为局部的优化，而变得更差。这个误区非常常见。

有效的度量应该区分外部度量和内部度量。其中，内部度量用以授权和赋能——授权给一线的团队，赋能它们分析哪里需要改进，怎么改进；而外部度量用作导向和要求——确保改进走在正确的方向上，并切实起到作用。



BizDevOps度量的目标是：

- 1) 赋能组织或团队掌握效能状况，分析效能问题，并针对性地改善能力和行为，从而提高交付效能，带来期望的业务结果；
- 2) 度量为改进行动的结果提供有效反馈。



在以上三个类指标中。业务指标因业务类型而异；行为和能力指标，需要基于具体的度量改进场景，选择和定制。本白皮书将只提供交付效能指标的参考建议。

分类	指标	描述	衡量对象
需求响应和交付的效能	业务需求交付时长	业务需求从被确认到完成完成交付的时长	业务需求
	业务需求的吞吐量	单位周期内交付的业务需求数量	业务需求
	业务需求的有效性	业务需求完成了所定义的目标的比例，此为参考指标	业务需求
	产品需求交付时长	从产品需求确认到完成部署的时长	产品需求
	产品需求的吞吐量	单位时间周期内，技术团队研发交付产品需求的数量	产品需求
	交付质量	缺陷数量、严重级别及分布，反映软件交付的质量	产品功能
技术和工程效能	部署频率	应用变更(生产代码的修改)部署到生产环境的频率	应用变更
	变更时长	从代码提交到成功在生产环境上运行的时长	应用变更
	服务恢复时长	服务发生线上故障之后，需要花费多长的时间恢复服务	应用变更
	变更失败率	多少百分比的应用变更会导致问题的发生(如服务降级、故障等)	应用变更

上表的交付效能指标分为需求交付效率和工程效率两个部分。需求交付效率体现组织顺畅和高质量交付价值的能力。工程方面参考了**DORA的度量模型**,反映组织的持续部署能力。光有指标,度量和改进还无法落地。一个完整可落地的度量体系需要包含:数据的采集、信息的获取、组织和呈现,以及如何应用度量来改进组织的运作和效能。接下来的一节,将涵盖这几个方面的内容。

• BizDevOps 的度量和改进体系

BizDevOps实施也是组织内部数字化转型过程,而度量则是数字化转型中的数据应用。我们可以从各类数据应用的案例中借鉴被一再验证的理念和实践。

在数据(尤其是大数据)应用中,DIKW是被最广泛采用的模型。我们也将基于它来设计BizDevOps度量体系。DIKW模型由数

据(Data)、信息(Information)、知识(Knowledge)、智慧(Wisdom)这四个层次构成,它们的首字母合在一起便是DIKW:

- **数据**:数据是在运作过程中产生的可以被采集的原始事实;
- **信息**:信息是基于数据生成的,并被赋予特定的意义或解释。信息可以表达为指标,也可以表现为图形和表格等其他形式;
- **知识**:知识是被有效组织的信息。有效组织的依据是,能够被用来系统回答重要的问题;
- **智慧**:智慧是应用知识指导有效的行动。从数据到信息再到知识,这些只能用来解释过去,只有智慧是改变未来的。

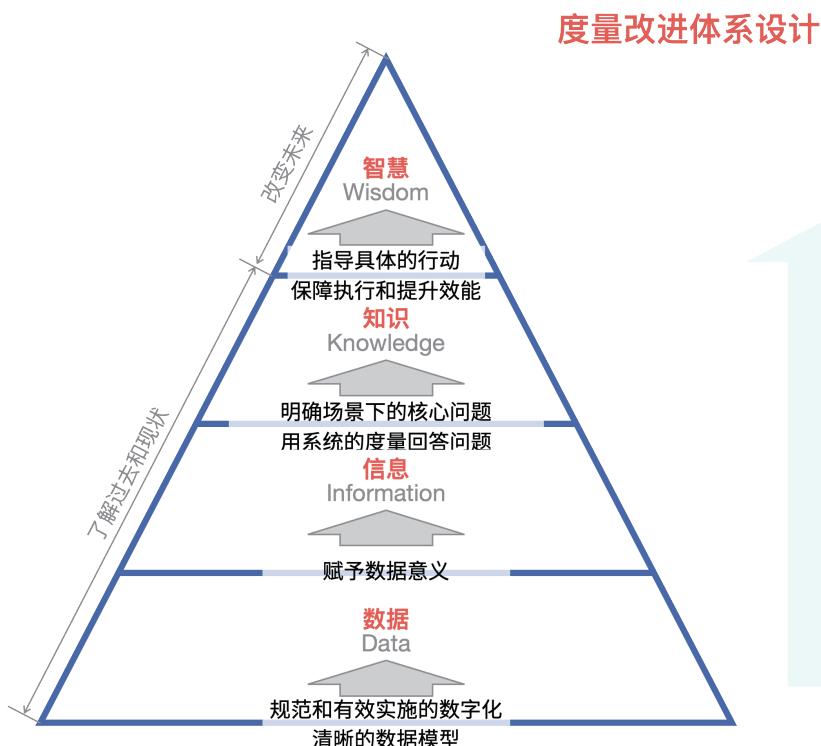


图3-17:BizDevOps的度量和改进体系

度量本身不是目的，度量的真正目的在于改变未来，也就是DIKW中“W”（智慧, Wisdom），但是智慧无法凭空产生，它需要数据、信息和知识的逐层支持。对应DIKW模型，有效的度量体系应该是：基于可靠的基础数据，组织有意义的信息，回答具体场景下的核心问题，指导有效行动并达成改进目标。



**有效的度量和改进体系应该是：
基于可靠的基础数据，组织有意义的信息，回答具体场景下的核心问题，指导有效行动并达成改进目标。**



可靠的数据、有意义的信息、回答核心问题的知识、带来有效行动的智慧，这是成功的度量和改进体系的4个要素，也是度量和改进产生的顺序。不过，在应用度量进行改进时，顺序正好相反。为了应用度量进行改进：

第一，要弄清楚，我们期望带来的改变——需要怎样的智慧；

第二，为了指导有效的行动，我们需要回答怎样的核心问题——建立什么知识；

第三，应该怎样系统回答这个问题——如何组织信息；

第四，怎样产生这些信息——采集哪些数据，如何保证有这些数据，并且数据是可靠的。

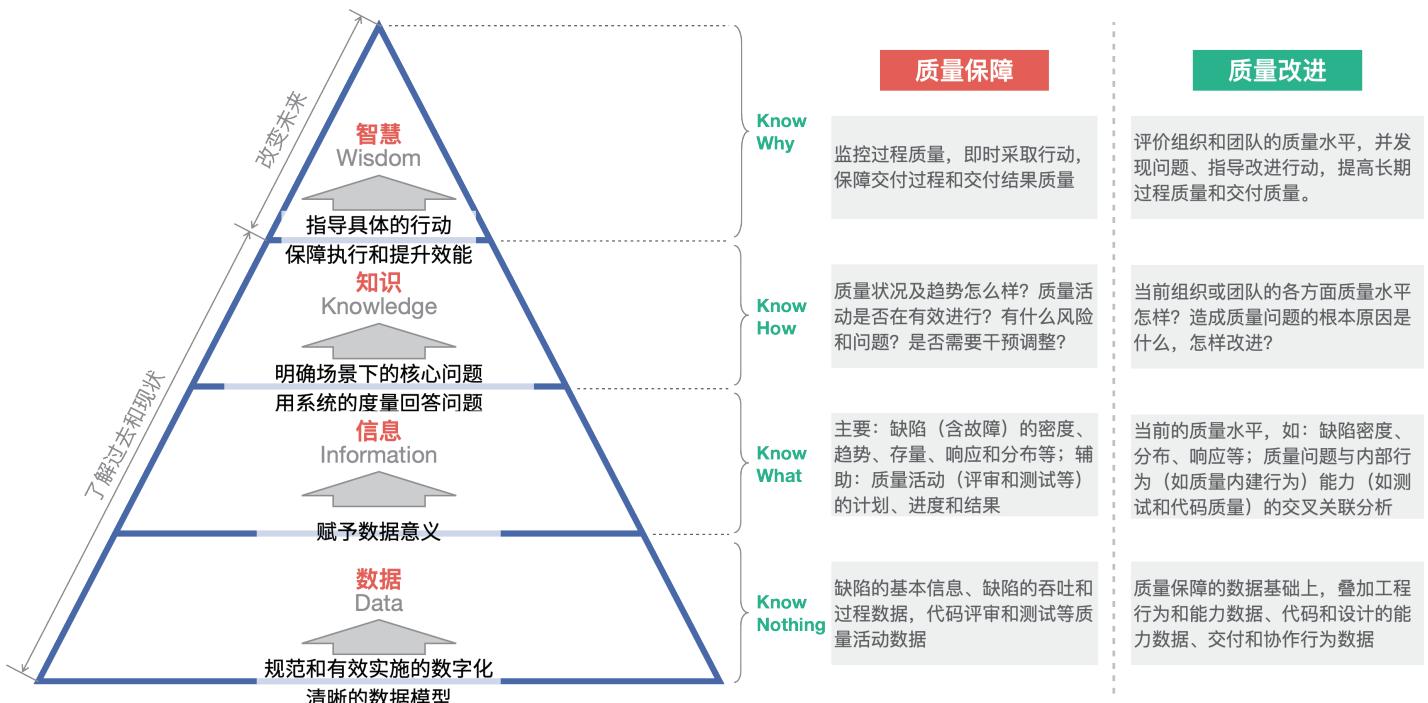


图3-18:应用度量指导改进的例子

上图给出两个应用度量进行改进的场景化案例，分别是质量保障场景和质量改进场景。两个场景目标不同，期望带来的行为不同，对应的由知识、信息和数据构成的度量体系也不同。

1. 智慧：BizDevOps度量和改进的目标

度量不是目的，度量的真正目的在于改变未来。BizDevOps度量期望改变包括近期的未来和远期的未来。改变近期的未来，靠的是透明现状和问题，保障更好的规划和执行。上例中，质量保障场景就属于此类。改变远期的未来，需要了解当前的效能水平，并分析深层次的问题。上例中，质量改进场景就属于此类。

2. 知识：为了支持目标的达成，需要回答的问题

明确度量的场景目标，接下来要确定的是，为了支持这个场景目标，我们希望通过度量回答什么问题。比如：为了保障执行过程的质量，我们需要回答，质量状况及趋势怎么样？质量活动是否在有效进行？有什么风险和问题？是否需要干预调整？为了改进长期的质量，我们要回答的问题是，当前组织或团队的各方面质量水平怎样？造成质量问题的根本原因是什么，怎样改进？

3. 信息：为了回答这些问题，可以应用的指标和图表等内容

确定了要回答的问题，接下来就是要组织信息来回答这些问题。比如：为了回答上述质量保障场景中的问题，我们需要的信息有：缺陷的密度、趋势、存量、响应和分布，以及质量活动（评审和测试等）的计划、进度和结果等。

4. 数据：产生需要信息的原始数据

离开可靠的数据，一切度量和改进都是空中楼阁。可靠的数据具备3个特征：1) 全量：记录数字化运作过程中所有的行为；2) 全要素：数据能体现各个需要的维度，并产生正确的关联，如缺陷与对应的代码关联，工程行为与需求关联；3) 实时：数据应该在行为产生的第一时间（如：需求分配，代码提交，部署发生等）作为副产物被自然记录，而不是后续专门补充。后补的数据既容易缺失，也会失真，更无法支持实时的应用。



如何才能保障数据是全量、全要素和实时的呢？数据不是一个独立的体系，它依赖于数字化的实施。在第二章中，我们介绍了BizDevOps的概念模型，它在其抽象层次上囊括了主要的概念，建立了这些概念的关联，并识别了核心作业对象。基于它扩展、定义和落地数字化体系，实时记录作业对象的操作记录，并关联相关对象的数据，可以保障数据的全量、全要素和实时。BizDevOps概念模型的一个应用场景就是指导数据体系的设计。



基于DIKW模型从场景目标出发，设计和应用度量体系，并落地具体的改进行动，前面介绍的其他BizDevOps实践为改进行动提供了参考，改进的结果最终应该在交付效能或业务结果中被检验。如此，就形成了持续改进和反馈的闭环，通过它进化组织的BizDevOps实践体系，从而加速数字业务发展和引领数字业务创新。

必致 (BizDevOps) —— 实践案例

案例一：迈向BizDevOps的招行银行精益管理体系

• 精益管理体系演进历程

精益管理体系紧紧围绕招行发展战略，同时参考业界先进的研发管理方法论，结合自身实践融会贯通，推动落地。

回顾招行精益管理体系的演进历程，主要分为四个阶段：

第一阶段(2008年-2013年)：“一体两翼”

- **关键词：**迈向规范化、CMMI、提升软件开发过程成熟度

- **举措：**在组织级层面引入CMMI模型，成立EPG(Engineering Process Group) 过程改进工作组，组建QA队伍，建立全生命周期的过程管理体系和过程资产库，建立协同工作机制和度量分析平台，重点规范研发过程和提升开发质量。

第二阶段(2014年-2017年)：“轻型银行”

- **关键词：**敏捷化、轻型化、CMMI+看板+敏捷方法+DevOps

- **举措：**在CMMI基础上，引入敏捷开发模式、看板、持续集成、持续交付等多种XP实践和DevOps实践，探索研究精益开发模式。在这一阶段，核心的关键是如何让IT自身能力和研发效能获得较大提升，练好IT内功。

第三阶段(2018年-2020年)：“Fintech战略”

- **关键词：**价值驱动的精益转型、CMMI+价值驱动的精益管理+精益之星平台

- **举措：**引入精益思想，建立业务、IT“以客户价值为核心”的统一价值观和价值流，构建端到端一站式协同工作平台，形成价值驱动的端到端的精益管理体系，助力招行Fintech战略落地与数字化转型。这一阶段，除持续提升IT研发效能外，更重要的，根据招行数字产品能力模型，全面提升数字产品治理能力，围绕“高质高效交付业务价值”，IT左移往前站，与业务深度融合，改变来料加工式的串行模式，形成价值驱动的精益产品研发新局面。

第四阶段(2021年-至今)：“3.0模式”

- **关键词：**大财富管理的业务模式、数字化的运营模式、开放融合的组织模式；迈进原生云时代

- **举措：**对齐业务战略和目标，打破组织壁垒，业务、开发、测试、运维、管理等高效协作、开放融合、责任共担，面向价值持续交付。随着招行原生云架构转型，继续深化价值驱动的精益转型，以产品思维为导向，纵向狠抓精益实践落地的有效性，横向狠抓端到端的数字产品治理能力与价值创造，助力招行“3.0模式”落地。

• BizDevOps的落地演进

回顾1.2章关于BizDevOps的定义，简要概括一下招行整体的落地情况：

1个总体目标：业务和技术有机融合、高效运作，赋能数字业务的持续创新和长期发展。

“价值驱动的精益管理框架”终极目标是对齐业务发展战略，合理投入资源，快速交付高价值。

3个能力要求：

以客户价值为核心的协同能力

从2016年开始,逐步推行DevOps,逐步打通Dev到Test再到Ops的价值交付链;从2020年开始,逐步延伸到Biz端,初步打通从业务开始到业务结束的完整链路和反馈闭环。

全链路的数字化运作能力

从2008年开始,研发体系的落地都伴随着IT系统的支持,特别是2014年开始引入精益实践后,自动化、线上化,将体系流程固化到工具里是整个IT部门的核心共识。遵循在关键节点上的核心流程、管理工具自研,产品化较好且有一定技术门槛的工具外购+定制的原则,确保从业务价值发掘、专题特性分析、版本规划、版本交付,上线后运营反馈等全链路的数据全量、全要素和实时的记录、串联和呈现出来。

基于高可用数据的过程透明和效能度量能力

从2008年开始,度量平台是一直持续完善的系统之一,只是在初期,很多数据都靠基层组织的度量管理员手工收集,但是其实时性和准确性受到很大的约束。随着近年来落地DevOps的相关实践,研发过程沉淀了大量的管理数据和工程数据,工程数据主要依托代码仓库、流水线、发布投产等工具的沉淀;管理数据主要依托看板、流程系统在每日的研发过程中沉淀。通过持续收集、可视化相关的数据,结合着每年研发过程改进整体目标,驱动整个组织的流程和效能持续改进和提升。

5个关键实践：**实践一:产品导向的团队组织方式**

2008年开始,基于CMMI建立的过程管理体系,主要以项目的方式进行交付。随着2014年开始尝试敏捷、精益、DevOps,以及Gartner提出了双模研发的概念,招行也开始在局部试点敏捷的研发模式,也算是产品导向的团队组织方式的雏形。随着试点的深入,结合银行的不同业务场景的特点,发现敏捷模式不完全适合招行的实际情况,部分需求的不连续性,也容易造成资源的锁定和板结。另一方面,随着DevOps实践的不断应用,基于系统的CI/CD工程实践,配合看板方法在基层开发组(最小的交付团队形式,类似亚马逊的2 pizza team)的推广,招行的研发体系逐步发展成产品导向的团队组织形式,只是在交付过程中,可以采用瀑布和类似SCRUM的方式。2018年,随着精益模式的逐步成型,主要包括精益项目模式和敏捷产品模式,虽然还保留着“项目”的外壳,但是在招行语境下,主要是交付周期和开发测试协作模式的一种方式,这两种模式还是会持续关注产品的长期演进以及资产沉淀和应用,管理产品实际价值的达成。

实践二:业务驱动的组织协同机制

2008年开始,基于CMMI建立的过程管理体系,主要以项目的方式进行交付。随着2014年开始尝试敏捷、精益、DevOps,以及Gartner提出了双模研发的概念,招行也开始在局部试点敏捷的研发模式,也算是产品导向的团队组织方式的雏形。随着试点的深入,结合银行的不同业务场景的特点,发现敏捷模式不完全适合招行的实际情况,部分需求的不连续性,也容易造成资源的锁定和

板结。另一方面，随着DevOps实践的不断应用，基于系统的CI/CD工程实践，配合看板方法在基层开发组（最小的交付团队形式，类似亚马逊的2 pizza team）的推广，招行的研发体系逐步发展成产品导向的团队组织形式，只是在交付过程中，可以采用瀑布和类似SCRUM的方式。2018年，随着精益模式的逐步成型，主要包括精益项目模式和敏捷产品模式，虽然还保留着“项目”的外壳，但是在招行语境下，主要是交付周期和开发测试协作模式的一种方式，这两种模式还是会持续关注产品的长期演进以及资产沉淀和应用，管理产品实际价值的达成。

实践三：应用为核心的研发资产和流程管理

在推广CI/CD工程实践过程中，招行建立了“系统-子系统-发布单元-服务单元”的基础术语模型，发布单元可以简单理解为一个微服务或者一个容器镜像，服务单元在发布单元的基础上加入了运维属性和环境等属性。代码仓库、流水线和制品仓库都围绕相关概念展开。通过应用这个基础术语模型，招行整体的CI/CD飞速的推广，也为敏捷协作交付带来了助推力。

随着大规模上云的完成，微服务拆分过多以及带来的管理复杂性也随之而来。2022年，通过与OAM两大发起单位的密切交流，招行也开始试点基于OAM的云原生应用管理，通过标准化工作负载，对齐数字产品和应用，实现不同角色的关注点分离，一方面降低开发者认知负载，另一方面更好地基于应用进行业务连续性保障，最终更好地保障价值交付。

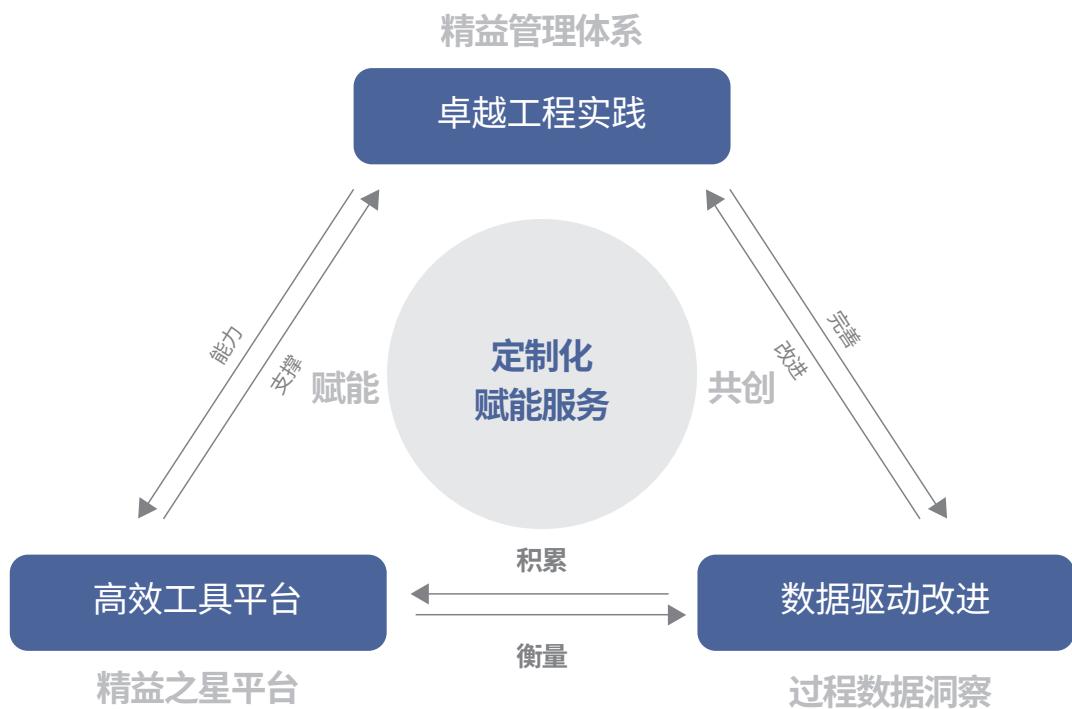
实践四：适配业务特征的持续业务交付

由于各个业务系统的发展阶段、技术架构、业务环境和成熟度不同，持续业务交付的形势也有所不同。实际运作过程中，招行也遇到了3.3.2章节中提到的几类问题：业务需求打包为版本交付，需要等待其他需求；应用变更需要打包在发布单元一起发布，一次发布包含多个应用变更，集成发布耗时过长。在理想模型下，每个业务版本、业务需求、发布单元、变更请求都是最小化的，但在实际适配的过程中，发现这些原则都对，具体情况下又难以操作：例如很多情况下自动化测试不足，需要由测试人员手工测试，这时候必要的整合和等待成为了平衡的结果。这个实践落地时冲突比较多、挑战比较大，需要交付团队以及更上层的领域团队一同持续优化。

实践五：全量、全要素、实施数据支持的度量和持续改进

伴随着BizDevOps工具链的逐步成熟，各级管理者和基层组织，对端到端过程中产生的各类数据也越来越感兴趣；另一方面，人员和服务数量的指数级增长，各类角色也对数据的实时性要求提出了更高的要求。招行的度量指标，主要从交付质量、交付能力、交付速度、持续交付、需求管理、资源管理、业务满意度和业务价值等多个维度进行度量，并开始基于部分重点角色和场景提供画像和洞察分析，赋能领域专家和一线管理者发现问题，持续改进。

- BizDevOps的未来之路



2022年，招行基本上完成了系统上云工作。展望2023年，围绕定制化赋能服务的打造，以下关键工作将有条不紊地展开。



管理体系要全面适配云开发的特点及要求，做好体系升级和过程保障，挖掘流程效率，促进科技组织的精细化管理水平提升；



工具平台要加速做好云时代的工具整合和适配，加强平台产品化能力和稳定性，强化质量门禁和安全扫描，提升BizDevOps端到端的用户体验；



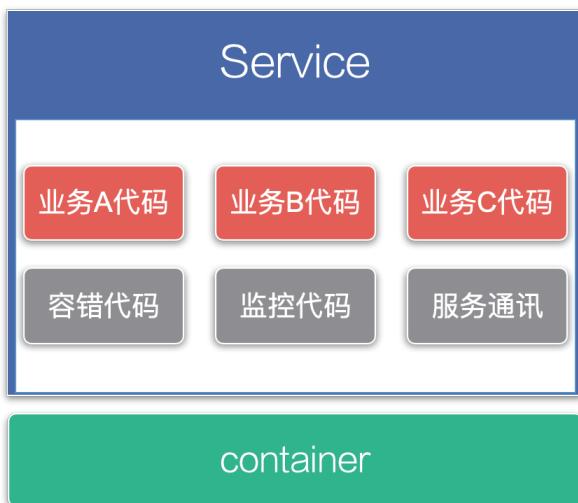
数据驱动方面，从响应式支持度量需求转变到研发过程数据体系化建设，升级研发数据中台，做好过程数据分析与洞察，提升研发过程数据价值。

案例2：阿里巴巴，以应用为核心打造持续业务交付能力

在阿里巴巴，以应用为核心的工程交付经过多年的发展，尤其是近两年云原生的推动，已经深入人心。我们以阿里巴巴某个电商中台部门的工程交付为例，介绍一下阿里应用为核心的工程交付实践。

• 业务特点和组织架构

中台的特点是业务方众多，需求来源众多，部署环境也非常多。对于一个中台的应用容器，它的内部结构可能是这样的，里面包含来自多个业务方的需求代码，同时又有来自横向中间件、SRE等团队提供的各类非功能性代码，这些代码共同构建组成了中台的服务容器。

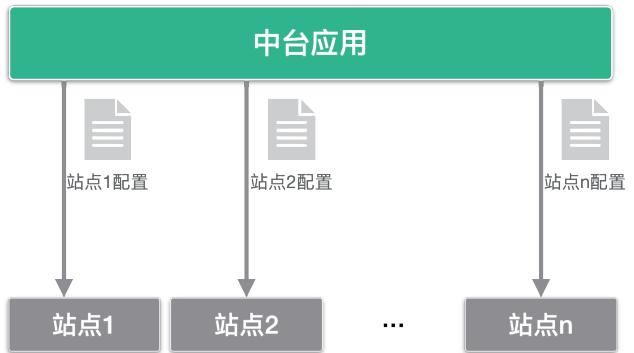


在技术架构上，业务研发并不允许直接修改中台应用代码，而是会以扩展点的方式，通过提供二方包给中台应用以集成业务能力。

另一方面，中台团队是按照应用划分的，每个应用有唯一的负责团队，这个团队负责该应用的开发、测试、部署和运维。因此，在组织架构上，类似下图所示。



在部署架构上，由于海外电商业务的特点，该应用会部署到多个站点，每个站点使用相同的部署镜像，但有不同的站点配置。这些配置包括：容器配置、弹性配置、中间件配置、业务配置等。



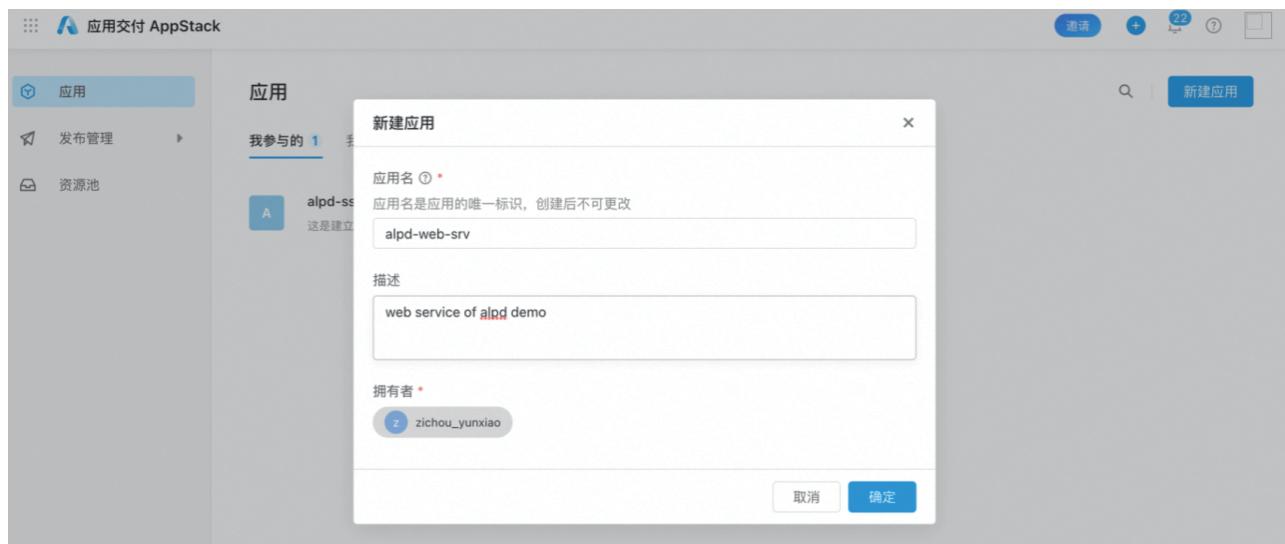
小结一下，对于中台应用，中台研发团队是唯一的负责人，负责维护该应用的全生命周期。业务团队是中台应用的贡献方，会与中台研发团队紧密协作。中间件、安全、云原生等团队是中台应用的依赖方，他们的变更也会影响到中台应用，双方也存在协作。

• 管理应用的研发资产

每个应用有唯一的负责团队，因此该团队天然有按应用聚合和管理研发资产的诉求，这里的研发资产包括：角色/权限、代码、配置、环境、部署编排等。

我们以阿里云云效平台为例，看一下应用是如何创建和管理研发资产的（为避免赘述，以代码和环境为例）。

创建应用



管理应用代码库

必致 (BizDevOps) 白皮书

管理应用环境



The screenshot shows the 'Environment' section of the BizDevOps application management interface. On the left, there's a sidebar with navigation items: 概览 (Overview), 研发流程 (Development Process), 变更 (Change), 流水线 (Pipeline), 环境 (Environment) [highlighted with a red box], 变量组 (Variable Group), and 版本 (Version). The main area is titled '环境' (Environment) and contains two sections: '测试环境' (Test Environment) and '生产环境' (Production Environment). Both sections show environment details like '环境级别/测试环境' (Environment Level/Test Environment) and '环境状态' (Environment Status). Buttons for '立即部署' (Deploy Now) are visible in both sections. A '新建环境' (Create New Environment) button is located in the top right corner.

应用是研发团队日常工作第一入口，也是资源的主要聚合和治理维度，还是团队工程效能度量和分析的主要视角。

- 对应于项目协作的项目，应用是研发团队的作业空间。当研发团队收到某个产品需求后，第一步就是找到需要变更的应用，建立应用变更，分配变更负责人。
- 财务或者基础资源团队进行开发测试所用物理资源盘点时，由于每个资源在CMDB中都标记了所属的应用，因此盘点和治理的时候，也会先聚合到应用，再有应用找到负责团队和更上层的组织。
- 在度量和改进工程效能时，由于应用是团队长期维护和演进的，基于应用可以得到团队研发相关的几乎所有活动事件，因此按应用进行度量可以做到全面、实时、客观。

• 定义应用的研发变更流程

在管理应用的研发资产(静态)的基础上，我们还会在应用上定义和管理应用的研发活动方式(动态)，具体表现为应用的研发变更流程。

我们认为应用的研发变更流程有如下几个特点：

1. 有明确的源头，常见的如某个代码库的变更分支，该中台应用要求流程从有变更开始。
2. 包含多个阶段，每个阶段有不同的分支、流程，且每个阶段可以独立执行，该中台应用包含开发、测试、预发和正式四个阶段。
3. 每个阶段有明确的准入和准出，该中台应用要求某个变更进入预发阶段前，必须通过测试阶段的验证且无安全问题。
4. 一个应用可以定义多个研发变更流程，该中台应用定义了标准流程和hotfix流程两个流程，分别用于一般日常需求的开发和紧急bug的修复。

一般情况下，只有应用负责人能够修改研发变更流程。

还是以云效为例，看下应用的研发变更流程。

在应用上创建研发变更流程

The screenshot shows the 'Application Settings' page for the 'alpd-web-srv' application. The left sidebar has a tree view with 'alpd-web-srv' selected, and 'Development Flow' is currently active. The main area is titled 'Development Flow' and describes it as a complete release process. It shows three stages: 'Development Stage' (Build, Deploy), 'Test Stage' (Build, Deploy), and 'Production Stage' (Build, Manual Checkpoint, Deploy). Below these stages is a section for 'Change Integration Mode' with a toggle switch.

配置阶段的变更卡点

This screenshot shows the same 'Application Settings' page for 'alpd-web-srv', but with more detailed configurations for the 'Development Flow'. Under 'Change Integration Mode', the 'Merge Change Feature Branches into Release Branch' option is selected. In the 'Merge Rules' section, the 'Merge Changes Manually via流水线' (Merge changes manually via pipeline) option is selected, and the 'master' branch is specified. The 'Access Rules' section shows a rule: 'Test Stage - Execution Result Equals Success'.

研发变更流程中所产生的各类事件和输出都可以作为变更的元数据，因此，可以基于这些元数据，进行应用的各类治理工

• 变更的开发、集成、交付

在该中台应用的研发实践中,开发任务等同于变更,对应用的任何变化,都是通过变更交付的。

中台应用的每个开发者都可以创建变更,每个变更在创建的时候都需要指定源头,变更的源头可以有多个,可以是需求,也可以是缺陷,无相关工作项的变更不允许创建的。

每个应用变更都对应一个变更分支,开发者可以基于该变更在应用平台上直接拉取新的隔离项目环境。一个隔离项目环境可以理解为该变更独占的一套完整开发测试环境,通过中间件技术既与其它环境在该应用调用链路上隔离,又能在调用依赖应用时连通。

开发者在变更分支上完成代码开发,触发变更的开发阶段执行构建、测试、部署项目环境等,验证通过后,该变更就可以在下一阶段被集成和验证了。

变更的集成验证一般由应用的测试负责,有些团队也会直接由开发进行集成验证。集成验证阶段通常会包含来自多个业务方的变更(以不同的二方包的方式)和中台应用自身的变

更。各个变更分支会自动合并,并进行验证。变更的验证大量使用了自动化测试的技术,会通过流量测试等自动化手段快速对应用进行集成验证,只有通过集成验证的变更才能进入交付阶段。如果有变更在集成验证阶段失败,它会从集成的变更列表中移除,通过验证的变更将会进入下一阶段。

变更的发布阶段一般由应用的发布负责人负责,符合发布标准的多个变更会被自动合入发布分支,进行构建和发布前验证,验证不通过的变更会被剔除,剩下的变更会重新执行发布阶段,直至完成所有生产环境的部署。

变更部署完成后,变更分支会自动合入主干,并关闭变更,同时通知变更关联的工作项。如果该工作项相关的变更都已经完成,可以设置该工作项的状态自动往下流转。

• 从业务看变更、从变更看业务

业务需求、产品需求、技术任务、发布、应用变更,以及变更相关的测试、部署操作,全都可以关联到一起并自动联动和更新。



对于业务团队,可以在业务需求上看到相关产品需求的进展,可以下钻某个产品需求,查看其相关应用变更的状态,及时发

现交付的瓶颈,并及早协作,从而做到从业务看技术。

• 基于业务看技术



站在业务的角度,可以从业务需求追溯与之相关的所有产品需求,以及为了实现这些产品需求,所涉及的所有应用变更及变更的发布情况。

对于研发团队,可以通过发布追溯到变更,从变更追溯相关的需求,以及为支撑该需求,所涉及的其它应用的变更。通过业

务需求到发布的端到端连通,可以有效管理,保证工程技术团队总是聚焦在最重要的业务价值上,避免浪费,也可以促使业、产、技三方有效协同,共同决策和改进。这也使得对业、产、技各方的价值评价得以统一到业务价值这一最重要的目标上,一方面可以基于业务看技术,另一方面可以基于技术看业务。



• 基于技术看业务

站在技术的角度,可以从变更请求追溯其实现的产品需求,以及这些产品需求所承载的业务需求。一个变更请求可以实现多个产品需求,同样,一个产品需求也可以服务于多个业务需求,通过这张关联视图,我们可以很容易地看到他们之间的关系及进展。

业、产、技的端到端连接和实时数据共享,还使得对业务的即时监控和反馈成为可能。通过为业务需求定义业务观测指标和预期目标,可以在工程发布的第一时间进行精准的业务监控和预警,并有效融合工程的灰度发布和业务的AB测试,使得技术在服务于业务价值的基础上,加速业务价值的验证,推动业务的创新。

附录

附录一：BizDevOps共促计划

BizDevOps 共促计划(英文名称：“BizDevOps Promoting Plan”，以下简称计划)，聚焦于BizDevOps体系的完善、应用与推广，加速企业和组织的数字化转型，为数字经济的发展贡献力量。是由南京大学软件研发效能实验室、北京极客邦科技有限公司、阿里云计算有限公司、思特沃克软件技术(北京)有限公司、招商银行股份有限公司、上海优川信息技术有限公司联合发起成立的专业性、全国性、非营利性的虚拟社会组织，并持续面向相关企事业单位、高等院校、科研院所、社团组织开放共建。

计划的宗旨是集聚产学研媒各方所长，探索验证BizDevOps理论、方法、技术和实践；定义BizDevOps的数字化模型与框架体系；打造和升级符合BizDevOps要求的基础设施和工具平台，宣传和普及BizDevOps的理念、方法和实践，建设BizDevOps文化；完善相关的基础和职业教育，培养BizDevOps人才。在数字经济时代，推动面向业技融合的数字企业治理模式。

附录二：参考资料

EDGE：价值驱动的数字化转型
机械工业出版社

阿里巴巴DevOps实践指南
<https://developer.aliyun.com/topic/devops>

价值流动：数字化场景下软件研发效能与业务敏捷的关键
清华大学出版社

2022 State of DevOps Report
<https://cloud.google.com/devops/state-of-devops/>

附录三：词汇表

BizDevOps(必致)

BizDevOps是数字化时代先进的产品交付及业务创新实践方法体系,它通过业务和技术有机融合和有效协同机制,打通从业务到开发再到运维和运营的价值交付链路和反馈闭环,并在此基础上实现全链路的数字化运作,保障并持续改进产品和业务交付的效率、质量和有效性,从而赋能数字业务的持续创新和长期发展。BizDevOps的中文名必致,取自“数字化转型,使命必达;业技融合,行稳致远”。

变更请求

为实现特定的产品需求或修复产品中的缺陷,某个应用所需要完成的工作。变更请求的实现,是包含设计、编码、提交和部署等在内的完整过程。与分散的技术任务不同,变更请求聚合了为响应特定产品需求,应用上所发生的所有技术活动。

成效指标(MoS)

专题的成效指标是专题在启动时确定的如何衡量其结果的标准。在具体落地时需要关注数据的可获取性,逐步形成自动化的获取、处理和呈现机制。

发布

指按照一定的流程,将某个发布单元(如产品和系统)中开发完毕的功能,分发到目标环境中,并使之对外部可见和可用的过程。

发布单元

是发布的最小粒度,它可能是单个应用,或有多个应用组成的产品和系统。发布单元下的变更请求需要一起集成、验证、部署后才

能共同发布,发布单元定义自己的集成发布流程,以保障发布的可控性和质量。

应用

应用是云原生环境下最基本的部署单元和运维对象,它由1到多个组件构成,运行在特定的基础设施之上,对外部提供服务或功能。应用拥有自己的代码、配置和数据,可以被单独变更、部署。应用之间应该是松耦合的。

专题

专题指的是业务和产品线为抓住业务机会而采取的具体行动。一个业务机会可能产生一个或多个专题,多个关联的业务机会也可能合并成为一个专题。专题与过去的研发项目有类似之处,但又有本质的不同。相对项目,专题更强调拥抱不确定性,基于专题的定位和成果定义,团队动态规划专题的具体内容,并在过程中持续迭代、反馈和调整。考核专题成功与否的并非是按期交付预先确定的内容,而是达成或超越所定义的成果,以及从中获取有效的产品和业务认知。

作业对象

价值交付链路上操作的基本价值单元,它:

- 1) 随时间发生状态迁移和流转;
- 2) 它的流转过程就是价值交付的过程。

作业空间

容纳并操作作业对象的空间,可以在其中定义作业的规则、流程、计划,提供或链接作业所需的资源,并完成价值交付的相关作业。



数字化转型
使命 必 达

业 技 融 合 行 稳 致 远