**从中台到平台(上)**

<https://mp.weixin.qq.com/s/dpkteHsQJ4Rwl6YNl2PVeg>

“平台化”是架构师经常用的一个词汇。什么样的设计实施可以称之为良好的实现的平台化，其实见仁见智。杨密总言道，我一般把平台发展划分为组件化、服务化、系统化。组件化是基础建设阶段，第二阶段就是服务化，由基础衍生到服务阶段，不区分业务或组件服务，为了后续的提速开发、规范、套用等。第三阶段则是微服务的成熟产出系统集成，产品输出，个人习惯称之”系统化”。 我理解此谓系统化就是平台化的一种表达。

**平台化的特点**

大略大家熟知的平台有SAAS平台、PAAS平台。比如阿里云的EDAS平台，定义如下：

EDAS 是一个围绕应用和微服务的PaaS平台，提供多样的应用发布能力和轻量级微服务解决方案，帮助用户解决在应用和服务管理过程中监控、诊断和高可用运维问题；提供开源软件Dubbo的商业版。  
关键字：应用运维、微服务、Dubbo、SpringBoot、鹰眼监控、服务治理

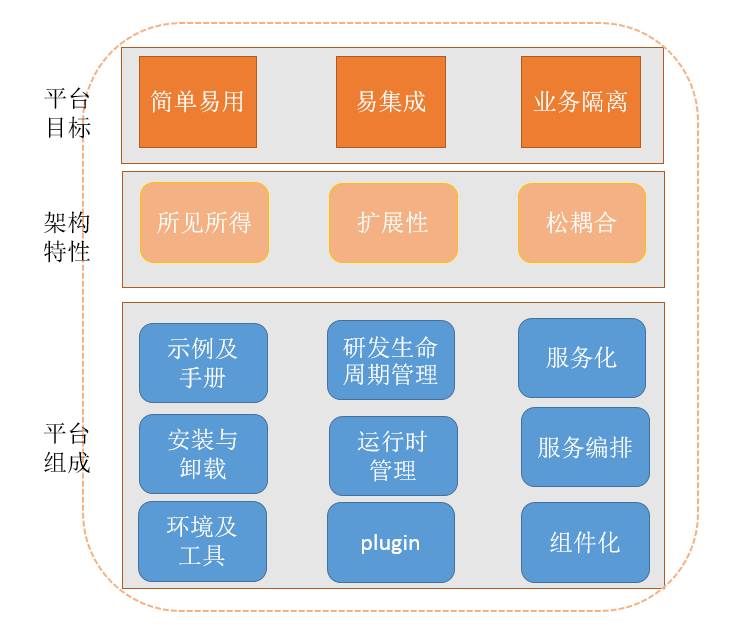


一个良好的平台，需要有确定的职责、内涵和外延。简单点说，就是做什么，不做什么；如何与其它平台协同。

Eclipse无疑是一个良好的平台，为无数开发者津津乐道，诸多企业研发产品都可以基于Eclipse做二次研发。基于 Eclipse 的应用程序的一个突出例子是 IBM Rational Software Architect，它构成了 IBM Java 开发工具系列的基础。

【平台化三部曲之一微核心可扩展架构 - 从Eclipse平台看交易平台化】一文的作者谈到，平台的开发，一般包括两方面的内容，一个是可复用性，一个是可配置型。Eclipse作为成功的开发工具集成平台，这两方面提供了很好的典范。从某种程度说，平台化就是领域平台化，也和SaaS化有很大的目标一致性，能够快速支持多业务的定制开发，同是保证业务之间互相隔离。比如交易平台化，你需要在一套交易系统中支持多种交易模式，多种业务实现。阿里的中台化战略就是建立在业务系统平台化基础上的协同。

总结，平台化架构的特点，简单易用、可扩展、业务隔离。如下图所示：

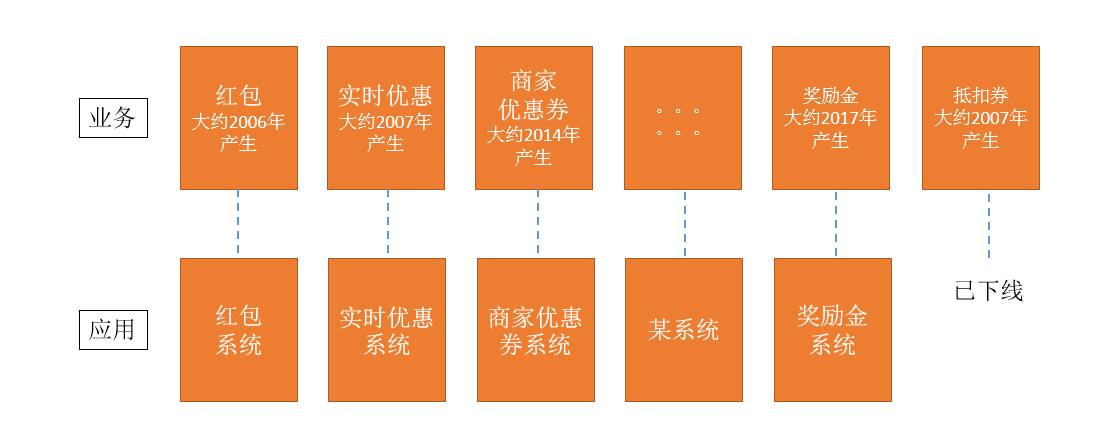


这里展开说一下，上图为示意，通过平台目标可以推导出架构特性，比如如何做到易集成呢？简单易用是不是可以进一步阐述为易安装、易配置、易运行？Eclipse则提供了良好的plugin机制来实现扩展性。一切都是插件。它自身就是一个基础平台和框架。

      以电商平台或者第三方支付的架构为例，进一步解释一下。比如支付平台从能力上提供了支付、退款甚至冻结、解冻等；那么支付平台对上层业务平台提供了服务化，组件化，甚至可视化编排的能力、研发各种工具、甚至支持扩展的plugin，示例、环境等。但研发工具、运行时管理、研发生命周期管理等可能由中间件团队、质量工具团队提供支撑。

**平台化架构的好处**

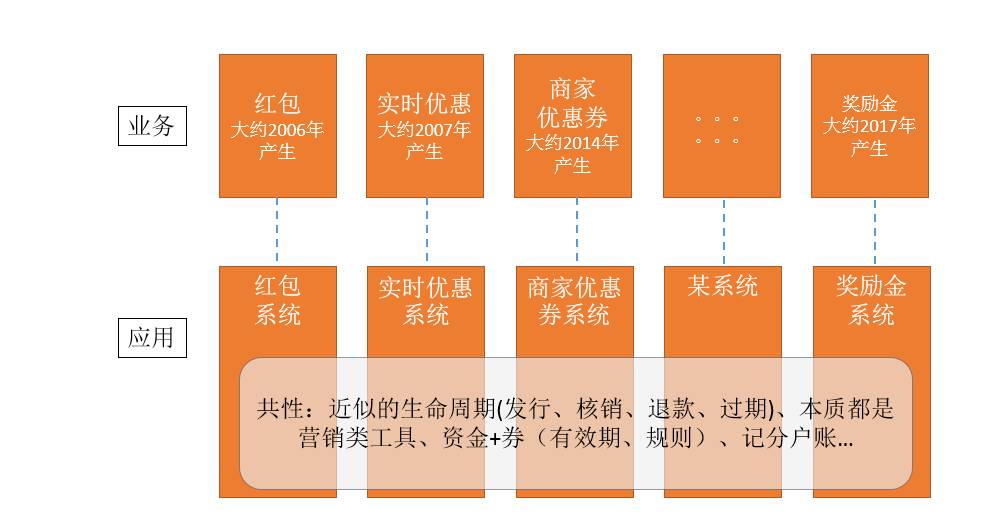
为什么会出现平台化架构，还得从烟囱型架构说起。回首我司营销工具的发展，百花争艳，变化万端。早期，很自然的是来一个业务，做一个（一套）系统去支持它。热火朝天干上一两个月，业务支持上了，但后续的维护是很大的问题。



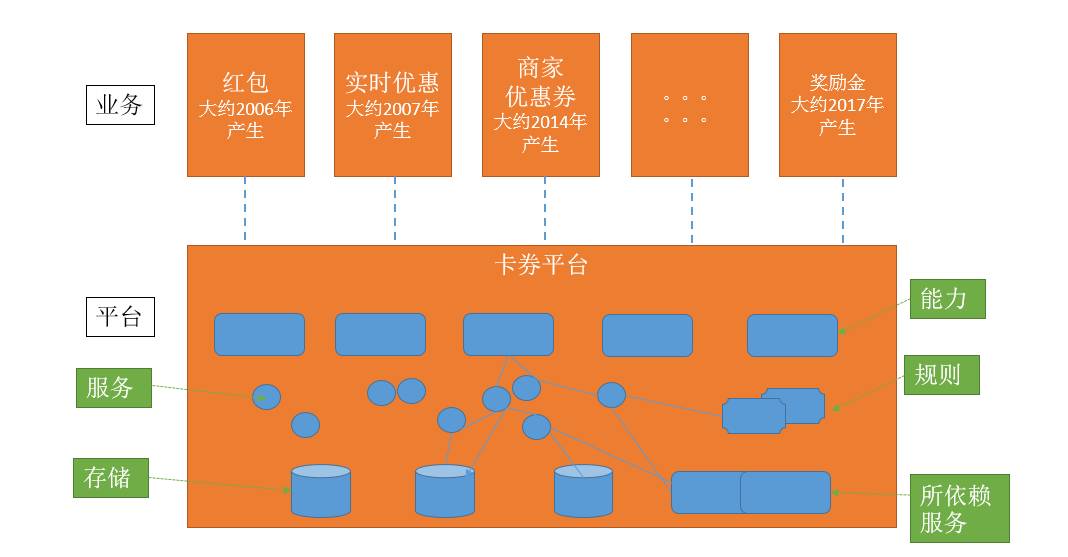
业内人士把见招拆招、垂直化发展、未做足够抽象通用的架构称之为烟囱型架构，烟囱型架构并非一无是处，在早期业务死活未知的情况下，不过度设计架构，能直接有效的支持到业务。谁手头没有死过几个业务呢？还记得我们团队有一个产品叫心愿单，刚刚上线就开始做2期研发，等到2期发布的第三天，组织决定这个产品要close了。那个产品的产品经理是我的好哥们叫邹衍，业务负责人是白鸦，就是现在有赞的创始人兼CEO。

不过，当业务发展起来之后，烟囱越树越多，成长的烦恼就如期而至了。第一个问题是人不够，业务响应慢了下来。我们以一个5人研发团队为例来说明一下这个问题。起初团队一个产品都没有，5个人1个月干出一个简单版本的红包系统；几年之后团队增加到10人，但手头要维护10个系统。那么平均人手一个系统，这时候，又来了2个新业务，团队派出3个人去干，大约要干4个月，严重不符合前端业务的响应预期。第二个问题是重复建设，同类烟囱系统中80%的功能是类似的，从数据库模型到主要业务逻辑，都是copy-paste加补丁，一步留神又踩到一个坑。第三个问题是维护成本高。日常升级包、咨询支持服务，团队疲惫不堪。基于此，80%甚至90%的共性问题，能不能抽象出来呢？核心领域模型是否可以是稳定的呢？从下图可以看出，这是可以做到的。

不过，当业务发展起来之后，烟囱越树越多，成长的烦恼就如期而至了。第一个问题是人不够，业务响应慢了下来。我们以一个5人研发团队为例来说明一下这个问题。起初团队一个产品都没有，5个人1个月干出一个简单版本的红包系统；几年之后团队增加到10人，但手头要维护10个系统。那么平均人手一个系统，这时候，又来了2个新业务，团队派出3个人去干，大约要干4个月，严重不符合前端业务的响应预期。第二个问题是重复建设，同类烟囱系统中80%的功能是类似的，从数据库模型到主要业务逻辑，都是copy-paste加补丁，一步留神又踩到一个坑。第三个问题是维护成本高。日常升级包、咨询支持服务，团队疲惫不堪。基于此，80%甚至90%的共性问题，能不能抽象出来呢？核心领域模型是否可以是稳定的呢？从下图可以看出，这是可以做到的。



在既要支持不断出现的各种业务，又要建设新平台的纠结权衡之后，我们启动了卡券平台化项目。建设路径是存量业务继续使用烟囱架构，但新业务随着新平台一起接入。第二步逐步迁移存量业务，实现烟囱系统的下线。如下图所示，卡券平台对前端业务提供统一的能力露出，由能力组装编排内部服务。研发规则运营、统一后台管理服务等。



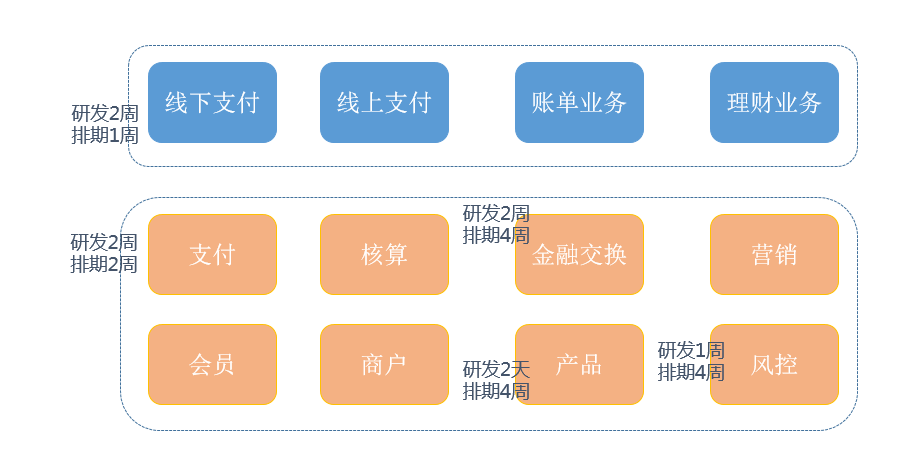
总结下来，平台化架构有以下好处，1是快速支撑、响应业务。2是抽象共性，边界清晰。快速支撑，响应业务是以终为始的出发点。架构如果不服务业务，再高大上都是扯淡。技术不是炫技，要服务商业。再谈谈抽象共性的问题，业务平台化要解决业务共性问题，比如天猫、淘宝都有各类营销活动。那么就抽象出一个营销平台来管理营销活动、营销工具的整个的生命周期管理。并提供给前端业务使用。

从组织层面，平台化会进行业务分治及归口管理，营销平台有对应的技术团队、产品团队和业务运营方。前端业务有需求发生的时候，有对应的受理和决策流程。

**平台化之后还有那些问题？**

平台化之后，按理说万事大吉了。但是又有新的问题出现。以电商平台为例，支付、会员、营销、产品都形成了对应的共享服务，但是对于一个新的前端业务，它要去理解各平台共享服务的职责，并协同，则并非易事。如果还涉及安全、合规、核算等，则涉及的团队更多。那么就有一个问题，已经搭了淘宝、天猫，要做聚划算需要多少人月？要做天猫国际需要多少人月？平台化的架构是尽量的走向了“各人自扫门前雪”，但对于创新的支持力度不足。互联网业务随时在试错，一些创新业务方期望是1-2周的期望上线，平台化发展动辄数月严重不能满足这样的诉求。

我们不妨去回顾那些在无数次发生过的似曾相识的场景，如下图所示：



一个线下支付的需求在对应研发团队是排期1周、研发2周，目标期望1个月完成上线。涉及到支付、金融交换、产品、风控等各团队，虽然都是1-2周的研发，但是实际整体上线可能是2个月。其间假使涉及到20个system，按最小单位1个system 1个开发1个测试算，涉及到40个人员的沟通复杂度，还没有算可能的运营人员，产品经理。同时在系统测试、联调阶段如果还遭遇若干环境问题，其效率可想而知。

总结，平台化架构由于缺乏对于前端业务一以贯之的端到端的支撑能力，平台与平台之间随时存在gap。平台化架构按照康威定律，必然是几个平台，几个团队，涉及到巨大的沟通成本而导致协作困难。平台化架构在数据化运营上存在短板，往往需要把多个平台的数据集成到一起并加工分析而产生新的支持到业务的价值。

**中台思想的提出**

马老师参观一个著名的游戏公司Supercell之后提出了中台思想，简言之就是“小前台、大中台”。 Supercell从表象看有200多名员工，一个游戏通常4-5个人研发，截止到2016年3月，Supercell面向全球市场推出了《部落冲突》、《卡通农场》、《海岛奇兵》和《皇室战争》四款游戏。大致可以分析Supercell采用的策略：

* 必须容忍很多失败：比如一个新项目在测试之前，团队就要设定一个指标，比如玩家留存、参与度，我们把这个目标告诉全公司的人，游戏进入测试之后，如果达不到指标，它就会被取消。
* 快速尝试：曾经在两年内，他们只发布了一款游戏《皇室战争》，但期间取消了9个项目，和若干很多优秀的创意原型。
* 招聘足够优秀的人：采用倒三角的模式组织团队，一个游戏公司等于若干创业小团队，小团队可以决定做什么，但没达成目标就必须中止。这决定了团队的每一个人是足够super的cell。

Supercell由于其游戏业务的特点，或与其它业务的研发模式不同。但有一个共同性思考，就是一个良好的中台首要的支持前台业务的快速创新。几个人干1-2个月，业务可以close，不用心疼。但如果百人月的产品，试错成本太高，时间方向也不满足高速变化的市场需要。

类比美军作战模式，就可进一步感受中台的作用。美军在二战时期，以军为单位作战；越战时变成以营为单位作战；中东战争时期进化为7人或11人的极小班排作战。之所以美军的“小前端”如此灵活，因为有强大的中台能力，给前端军队提供各种资源支持以及中台炮火群支持打击。

