携程运维工作流平台的演进之路

前言

随着互联网技术的迅速发展，运维的事务也日益复杂，如何能更加高效的协调好人、工具与流程之间的关系，把运维人员从低效率、高强度、易犯错的人工操作彻底解决出来，让他们的能力与精力有更大程度的发挥，将是一个很大的挑战。

携程运维工作流平台在经过三年时间的演进，从最开始引入商业产品BMC Remedy ARS做为底层单一工作流引擎，慢慢演化到抽象工作流平台的建设并扩展支持更多开源的工作流引擎，同时把原来的平台进一步分层、建立了统一标准的接口，对业务进行了服务标准化、业务流化以及流程自动化的改造。

本文将从以下几个主要方面分别阐述流程平台的建设与演进：

- 面临的挑战
- 运维工作流平台的建设
- 收获总结
- 下一代流程平台的设计
- 未来展望

一、面临的挑战

在讲挑战之前，我们先简单看一下携程运维工作流平台的演进历史：

流程平台的演进主要分为以下三个阶段：

第一阶段：摸索期

从2013年下半年到2014年上半年，我们引入了BMC Remedy ARS产品作为携程IT服务管理工具，但是我们发现它的基于工作流工作的思路可以借鉴到更复杂和核心的运维流程中，所以开始尝试使用它的底层引擎构建我们自己的流程。

第二阶段：成熟期

在2014年下半年到2016年上半年，在Remedy平台上相继开发了一系列的流程，包括：服务器上线、下线，应用上线、下线、扩容、缩容、迁移等等，还包括ENP，API gateway等一系列标准化的接口服务模块，开始渐渐形成流程平台。

第三阶段：革新升级期

在经过了一个成熟期之后，现有的流程也慢慢暴露出了一些新的问题，包括流程的可视化、价值数据的挖掘、底层流程引擎的单一，为了解决这些问题，我们从去年下半年开始，对原来的流程进行了重新设计，抽象出更加标准化的流程模型，以及标准规范的可视化和监控。

那么在没有流程平台之前，我们到底面临着怎么样的挑战呢？

在这里我给大家举一些运维过程中常见例子，比如日常工作，当用户碰到问题时IT部门报障时，用户如何跟踪问题处理的进度，又或者当网站发生故障时，如何能够快速恢复服务，后续的问题分析是否有流程支持，运维人员的日常变更操作，是否有过风险评估

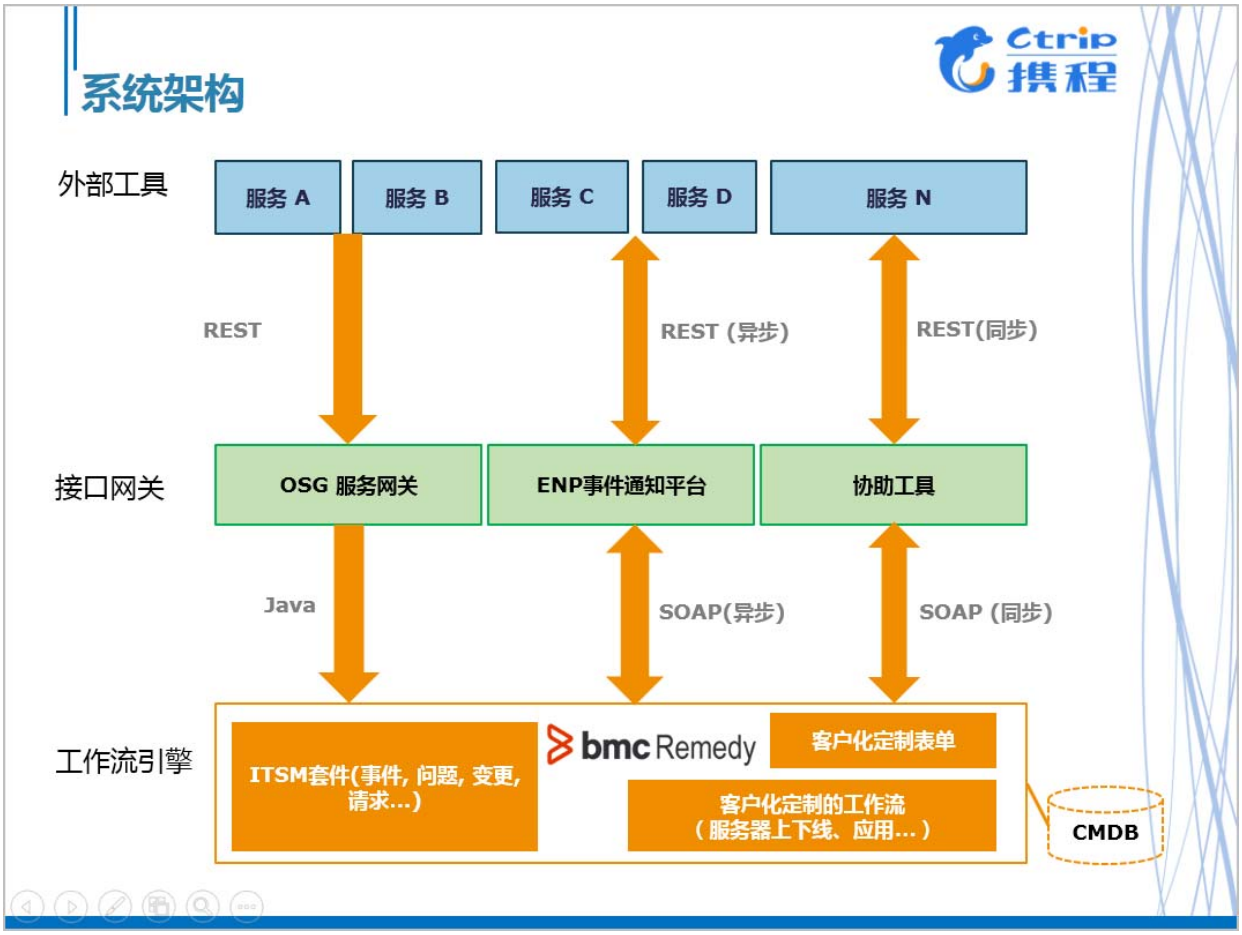
以及审批授权，我们又是如何保障配置数据的可靠、准确性，服务器如何上下线，应用生命周期有没有统一的管控，等等，正是在这样一个背景下，我们开始着手设计建设一个综合的工具平台来统一管理运维过程中涉及的这些流程。

二、工作流平台的建设

关于自动化流程平台建设，自动化的意义相信大家应该都是有共识的，在上面的挑战中我们已经提到了这些运维过程中大家可能会碰到的问题，那么我们是如何去建设这个流程平台的呢？

系统架构

我们先来看一下最初设计的流程平台系统架构。



从下往上看主要有三层：

底层工作流引擎

前面我们已经提到了，在一开始，我们引了BMC Remedy产品主要作为内部IT服务管理流程的支持，在原有的ITSM包括事件、问题、变更以及请求单基础上做了携程自己的一些客户化定制，同时在这个平台上设计开发了包括服务器上、下线，应用相关的上线、下线，扩容、缩容等支撑业务的相关流程。

中间接口网关层

在中间接口网关这层从左往后分别是OSG服务网关、ENP事件通知平台以及协助工具，为什么会有这样的设计呢，主要有以下几点考虑？

第一， 外部工具服务过多，需要统一标准化管理

需要对流程接口服务进行标准统一的管理，所有工具提供的服务需要注册在平台；

第二， 服务访问权限的控制

可以通过平台对提供服务进行权限控制，可以查看提供的服务由哪些外部用户或者工具调用，同时申请了哪些外部工具的服务，可以对申请进行授权。

第三， 工具提供的服务质量

可以通过可视化的前端页面查看当前提供的服务的质量，比如服务的响应时间。

第四， 访问日志

当进行问题分析排障时，可以有被追踪的日志。

第五， 产品局限

现有解决方案提供的接口只支持JAVA以及基于Web Services的SOAP协议，无法提供很好的扩展，所以需要在这层做些接口的包装，以支持更主流灵活一些接口协议，比如RESTful。

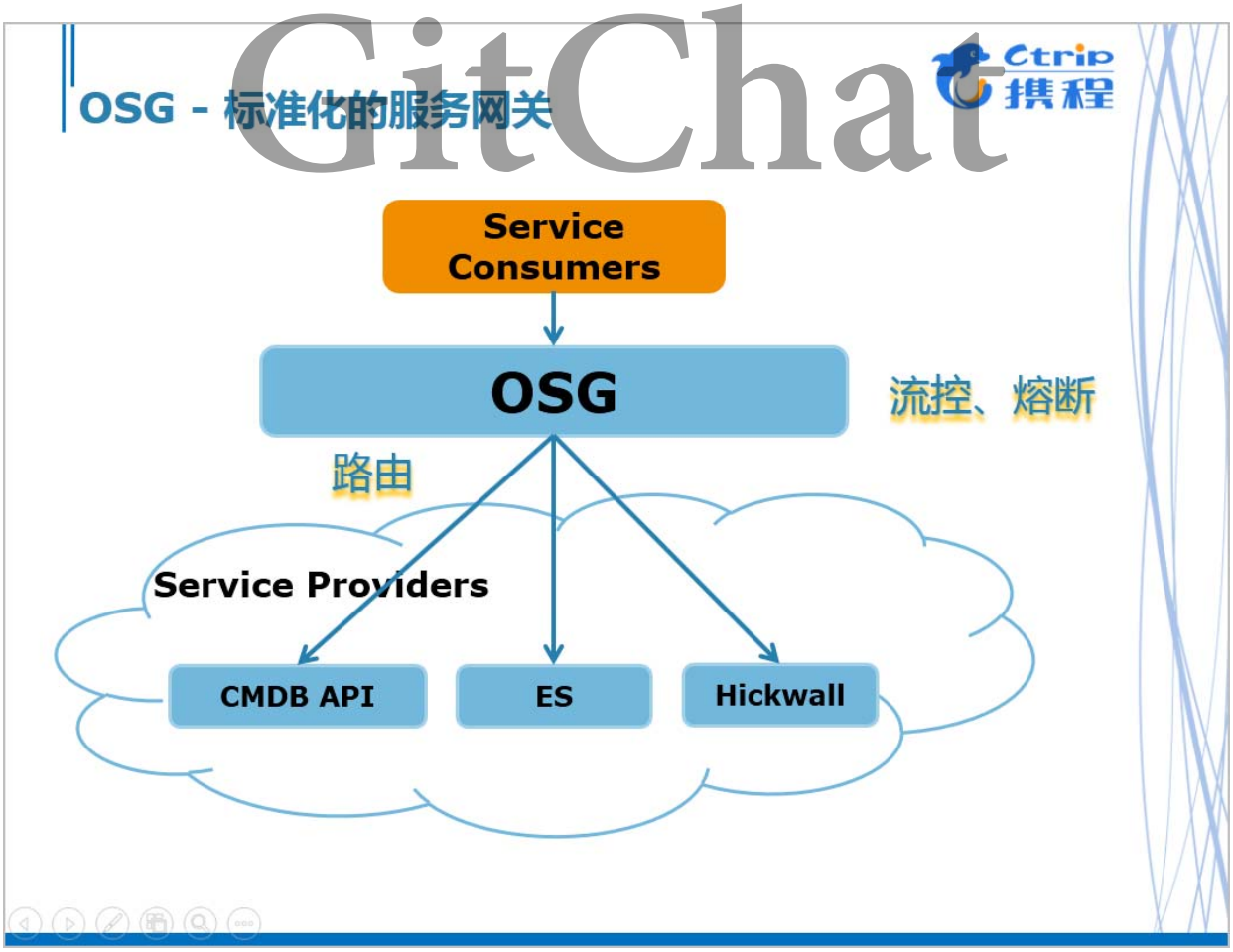
上层外部工具外服

最上面就是需要来访问流程的所有外部工具服务。

标准服务网关 – OSG

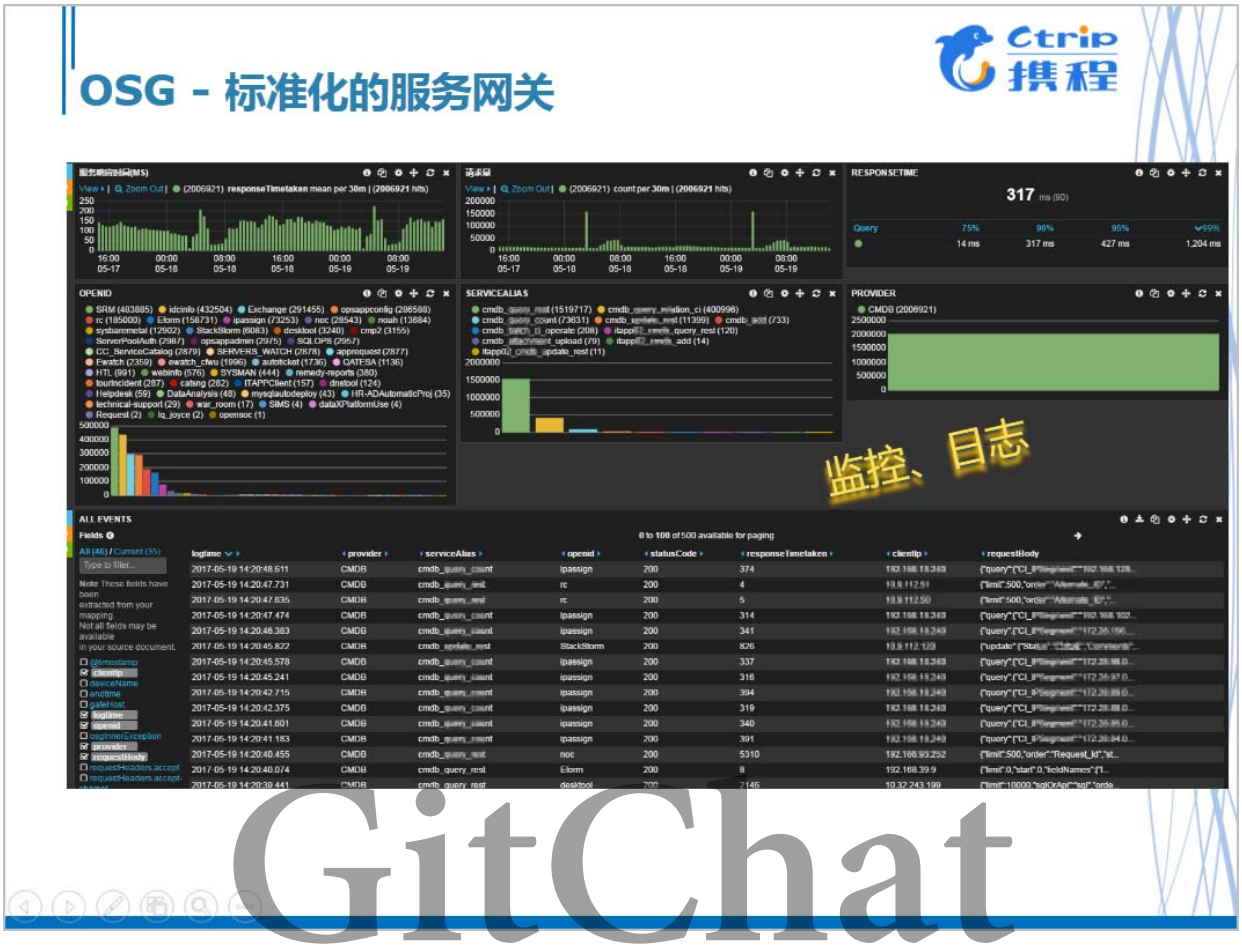
OSG是接口网关的核心的组件之一，那么什么是OSG呢？

OSG，全称：Open Service Gateway，开放的标准服务网关，外部工具作为服务提供者可以将他们的服务注册在OSG网关，其它工具若需要消费可以找到对应的服务，以服务消费者形式申请某项服务的访问权限。



当服务与服务之间相互交互的时候，OSG就充当着路由的角色，对访问的请求进行转发，OSG另一个主要功能就是流控熔断机制，可以为服务设置每分钟最大访问次数，当最大的访问次数超过设定阀值时，服务自动设置为Blocked，直到服务提供者重新启用。

对于工具之间服务相互访问日志都会被集中采集并吐到后台ES服务器，用户可以通过前端可视化的界面对服务的质量、工具的访问进行实时监控、同时可以根据日志进行问题排查。



流程与外部工具的桥梁 – ENP

ENP – Event Notification Platform（事件通知平台），这里大家可能会有这样的疑问，这个不就是消息队列吗？可以这么说，事件通知平台设计参考消息队列，但实现上又稍有些不同，之所以会有这个平台，是因为从Remedy产生的消息数据在处理上有些局限，所有的数据的格式化处理都需要交给ENP，由ENP根据设置规则进行封装处理后再转发给外部工具。

为什么需要这么一个平台呢？

刚开始Remedy平台上开发第一个服务器上线流程的时候，整个上线流程中涉及到了多个环节与外部不同工具交互，不同工具之间的接口实现非常复杂，所以在流程从半自动化到全自动化的转化过程中，开发人员花费了大量的时间与精力与工具进行联调，而且不同接口在实现方式也不统一，比如有些是基于Soap的Web Service，有些是RESTful API等等，另外通过这样一个平台可以降低工具之间的耦合度。

举个场景：

我有一个应用，调用了很多其它服务，其它服务有时会发生异常，为此又不想花太多精力去实现与维护。

我希望：

- 及时通知我
- 及时通知服务提供人
- 能看到异常发生频率、次数、时间
- 能看到异常的内容
- 能看到所有通知事件

那么平台是如何运作一个消息从产生到工具的传递呢？

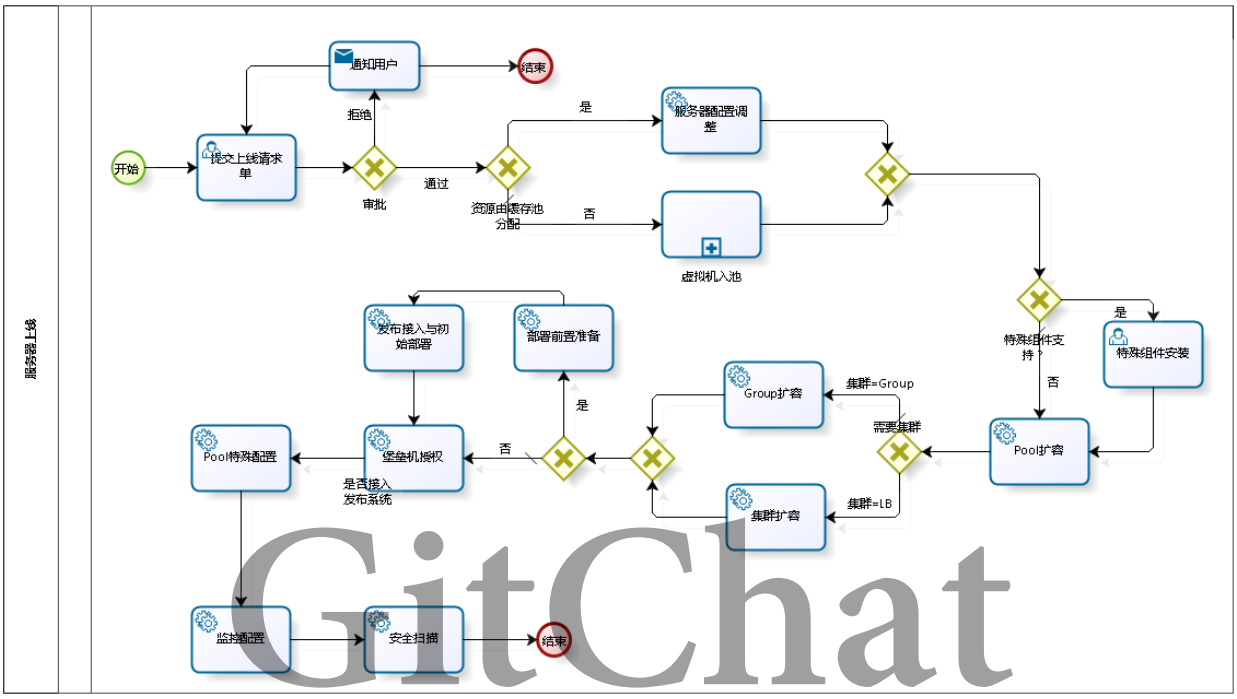
1. 首先需要在平台注册一个事件。
2. 工具访问之前需要在平台上搜索该事件到并且订阅，需要提供一些必要的信息，比如工具服务的调用地址。
3. ENP根据规则对消息进行格式化并封装。
4. ENP转发消息到外部工具。
5. 工具回写消息到ENP，ENP最终回写到流程平台。

流程场景：服务器上线

下面我以服务器上线流程具体场景为例进行说明如何设计流程：

在经过前期与各个业务部门，运维团队一起分析设计出来的最终服务器上线流程图，流程由一系列流对象组成，这些流对象可以任务，也可以是子流程，比如下图中所示的子流程“虚拟机入池”，同时这些任务与子流程之间由连接对象衔接，比如顺序，并行，分支判断处理。

流程还需要设定了一系列的业务规则，包括审批、状态迁移、SLA、CMDB数据落地等等。



可视化的流程运行实例

在有了流程之后，对于不同角色的用户，比如运维人员、开发人员、流程组以及管理人员，需要有个可视化的界面：

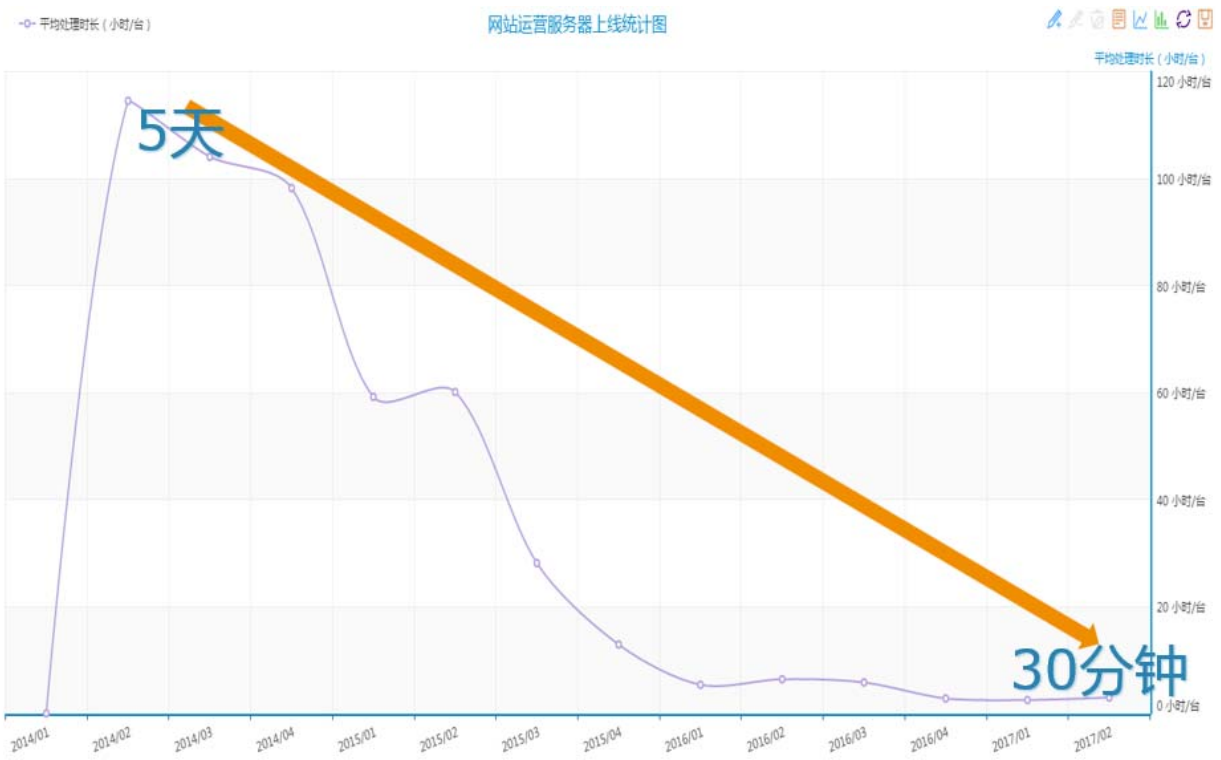
- 能清晰直观的看到所有运行的流程运行。
- 能清楚了解当前某个流程运行或者等待在哪一个环节。
- 流程的运行状态、时效。

下图就是流程平台可视化界面，可以看到目前所有运行中的流程实例，它们的运行状况，流程的运行时间，不同流程的运行数，当某个流程实例运行超过预定SLA时长，也可以查找超时子单，并找到相应的责任团队。

通过打开单条实例数据，可以进一度查看实例详情，在此可进一步看到所有运行或者已经完成的任务以及它们的处理时间，一旦任务处理异常挂起时，可以通过查看任务的责任团队迅速找到相关责任人进行快速处理。

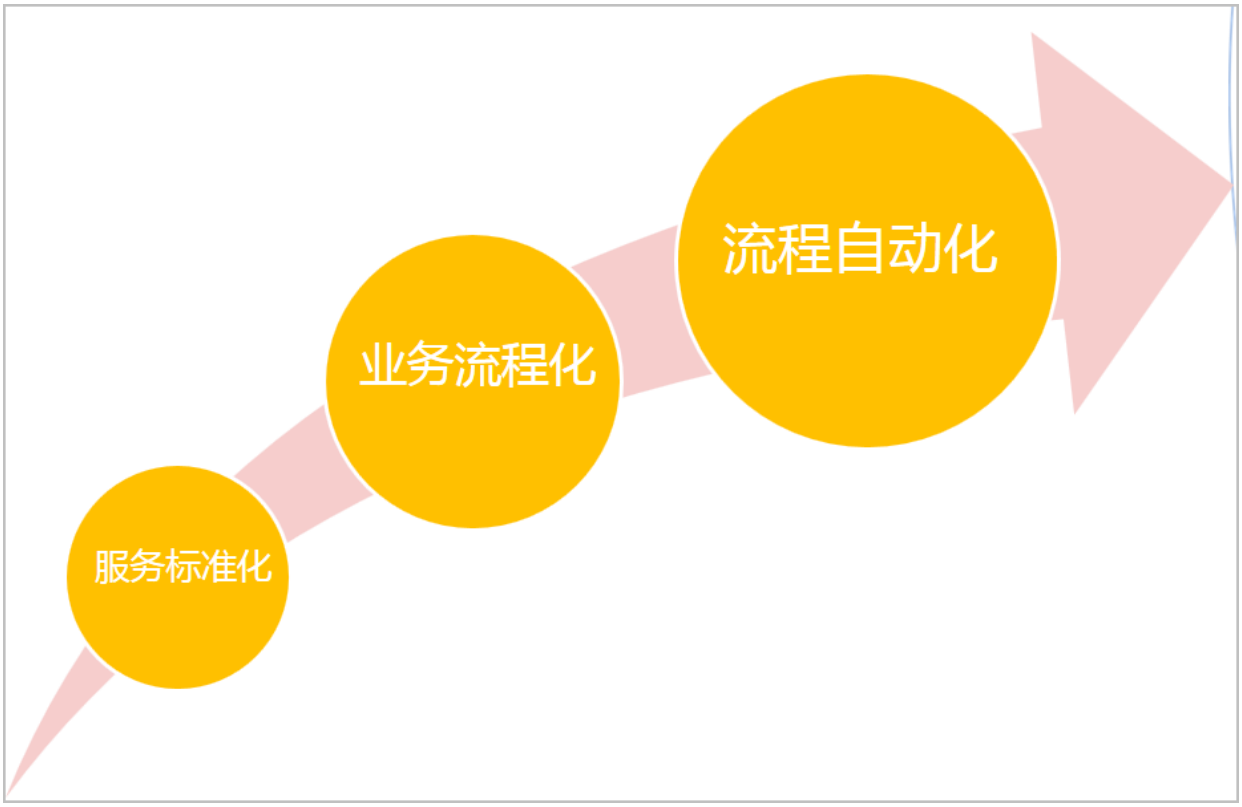
三、收获总结

10X服务交付能力的提升



2014年当时还没服务器上线流程时，业务部门从提出需求到最后交付平均需要一周左右的时间，最长可能会达到二周，所有上线的相关环节的协同操作几乎都由低效率的人工完成，很少有工具参与自动化处理，有一个场景让我至今映像深刻，当时整个网站运营中心的各个运维团队座位比较近，所有每当有上线的时候，经常会听到不同团队之间隔空对话方式进行沟通，还停留在通信基本靠吼的原始时代，在有了流程支持，这种低效率的团队合作方式也大大得到了改善，各个运维团队开始尝试开发自己的工具并接入到流程，流程也慢慢的从部份工具的接入的半自动化到所有工具接入的全自动化处理模式运行，在流程完全自动化后，最终上线效率得到了大大的提升。

服务流程标准化流程化到自动化



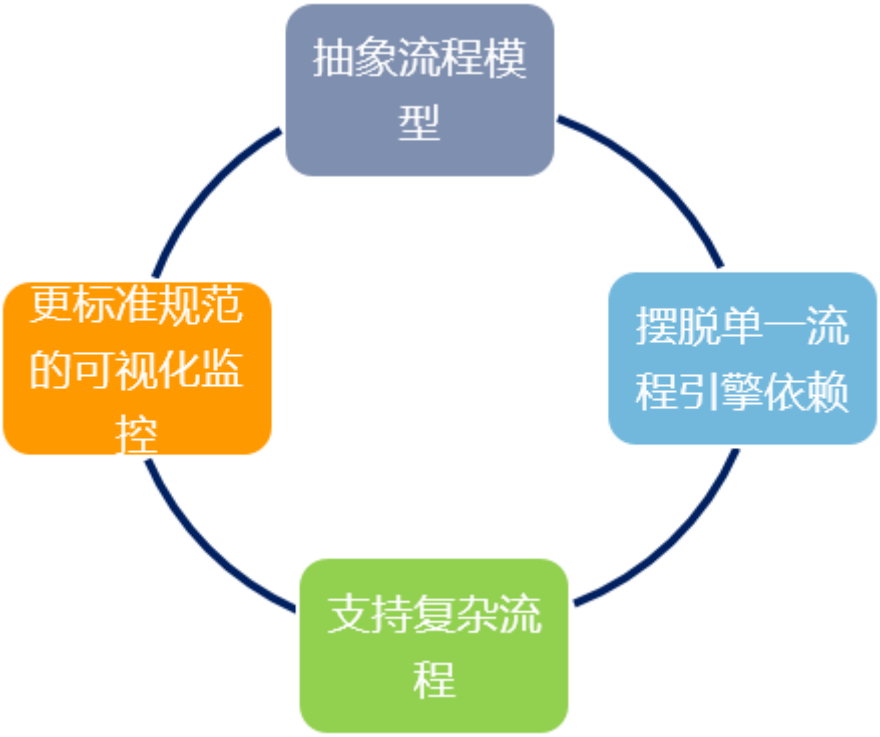
工作流平台的建设过程主要也是经过了以下几个阶段的发展，

第一，服务标准化，前面我们已经讲到了关于服务标准化，就是把现有的平台进一步分层，建立标准接口，像OSG这种的标准化接口网关。

第二，业务流程化，梳理各项业务，把大部分的IT运维工作流程化，确保这些工作都可重复。

第三，流程自动化，把运维人员从低效率、高强度、易犯错的人工操作彻底解决出来，让他们的能力与精力有更大程度的发挥。

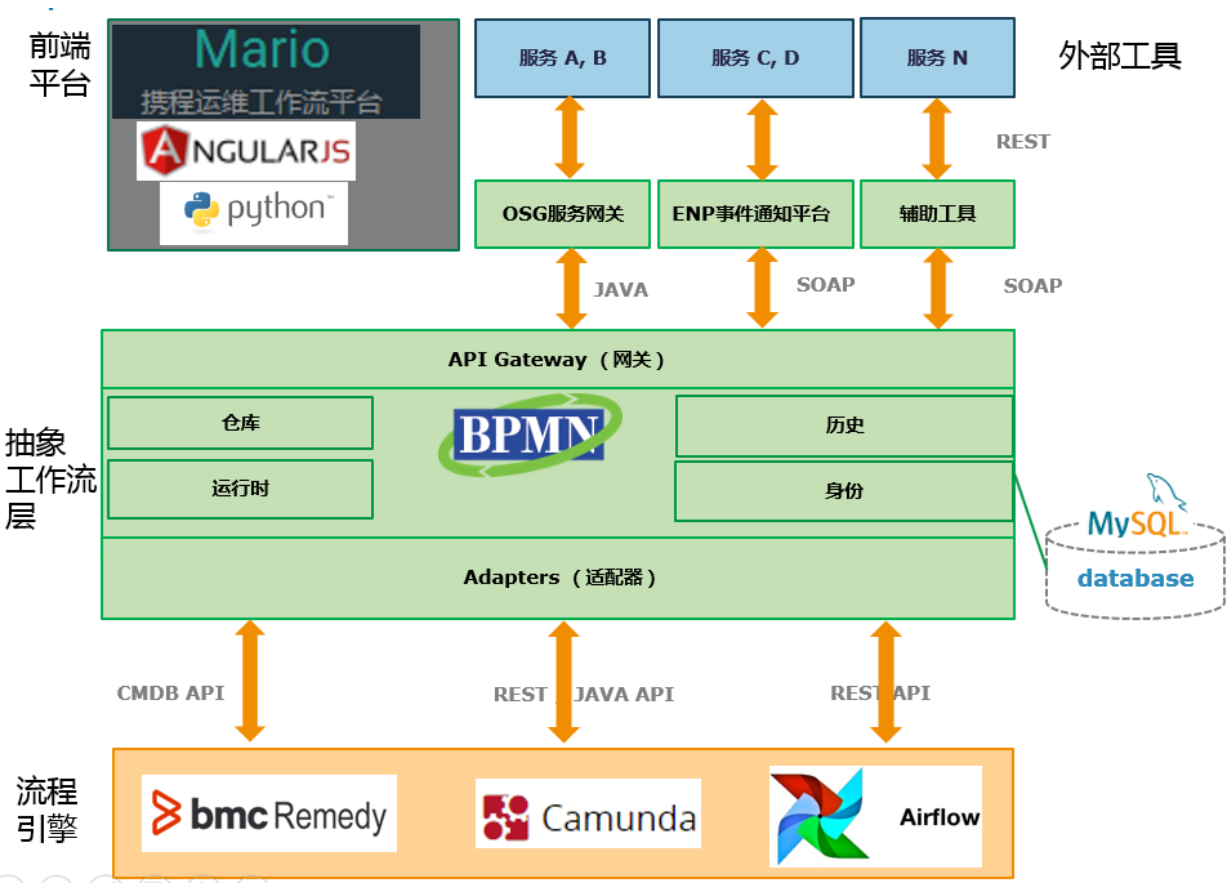
在经过成熟期后，现有的流程平台也逐渐暴露出一些问题，最主要的就是过度依赖单一的底层流程引擎，我们希望能够扩展更多的开源流程引擎，多个流程引擎之间可以随意切换，最终可以完全替换掉原有的商业引擎，如果需要支持更多不同的流程引擎，那么对流程进行抽象是必不可少的，建立出一个新的流程模型。另外一个问题，原来的流程引擎只能处理串行、并行以及简单的分支合流，新一代的引擎可以覆盖到所有复杂的流程分支处理，接下来的章节我们会讲到，如何设计处理复杂的场景。



四、下一代流程平台的设计

改进后的系统架构

我们再来看一下改进后的系统架构。



可以对比一下原来的系统架构，大家应该能发现他们之间的主要区别，底层除了原先的Remedy外，可以引入其它的开源流程引擎，比如基于BPMN标准的Camunda，或者Activiti、Airflow、Stackstorm等等，中间也是最核心的就是抽象工作流层，主要以几个模块组成：

1. 适配器

   主要负责不同流程引擎之间的数据映射交互。

2. 基于BPMN标准的模块，包括仓库、运行时、历史、身份

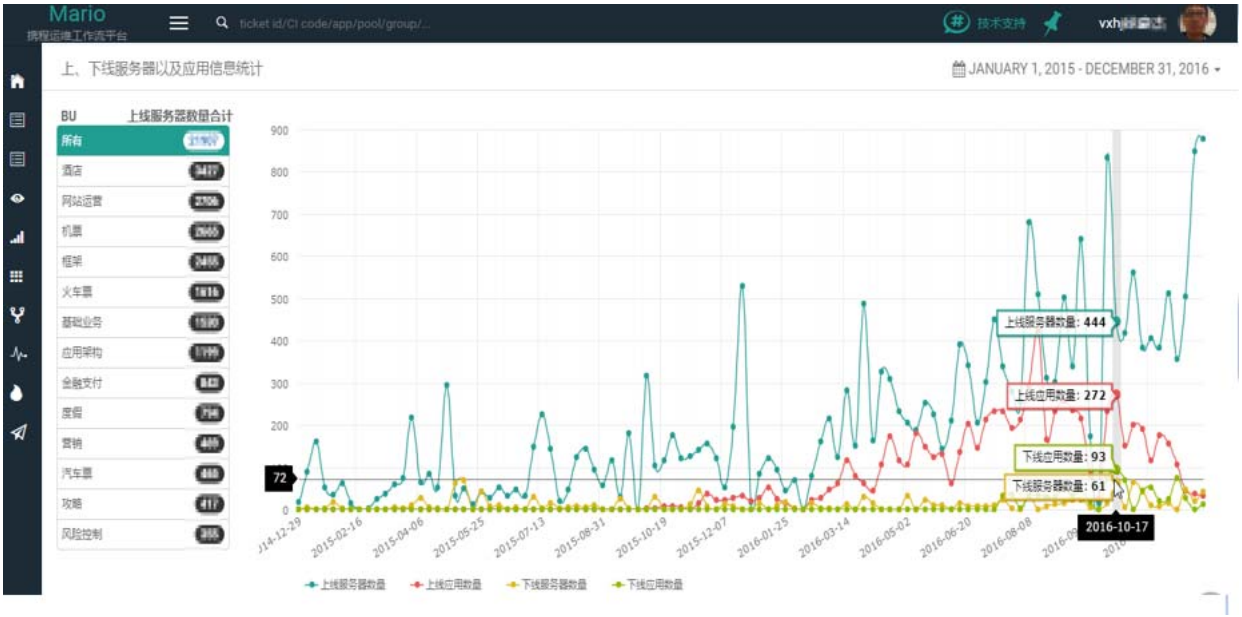
   仓库（Repository）主要存储流程的定义以及部署信息，所有流程执行过程中产生的实例、任务实例以及变量实例的数据会存储在历史（History）模块中，运行时（Runtime）模块主要负责所有正在运行中的用户任务，执行实例以及定时作业等，最后身份（ID）模块是用于存储用户、组以及他们之间的关系。

3. 接口网关

   主要负责与原来的接口层进行数据之间的交互，并实现与外部工具服务的通信前端由自研的Mario运维工作流平台做为核心数据的可视化集中展现，监控报表等等。

可视化

除了原来对流程实例以及实例详情进行可视化管理外，底层的其它有价值数据并没有得到深挖，比如跟业务相关的数据，各个业务部门关心的历史服务器、应用的趋势等，那么通过新的可视化我们可以更清晰看到业务部门的相关指标数据以及历史趋势。



监控告警-EITS

流程执行过程的异常由监控告警平台EITS负责处理，通过这个平台我们可以配置告警策略以及不同报警的分类，一旦流程执行异常时，告警会通过邮件方式提醒用户处理，用户可以登录到平台查找到对应的告警进行问题处理。

支持复杂流程

前面我们提到了，原来的流程引擎只能处理串行、并行以及简单的分支合并，在新一代流程平台我们做了一些设计调整，可以覆盖所有流程场景，我们以下图中最后面的一张图为例进行详细说明：

1. 任务a1处理完毕后，同时产生三个并行的分支任务

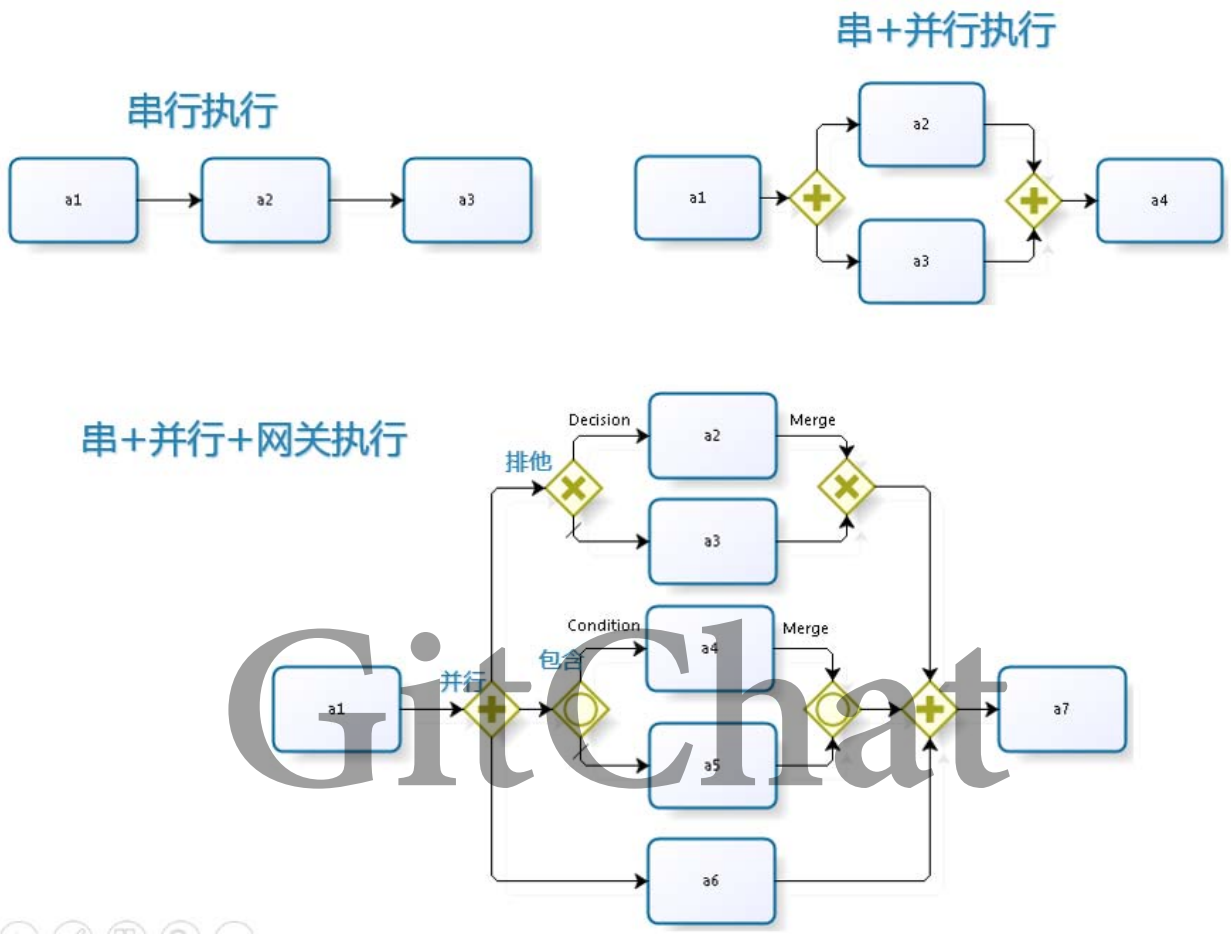
2. 分支一由排他性的网关处理，根据中间产生的不同数据进行条件判断，如果满足条件则执行任务a2，若不满足，则执行a3，排他性网关只会有一个任务被执行

3. 分支二由包含性的网关处理，任务a4 如果满足条件则被执行，a5始终会被执行

4. 分支三只不包括任务网关，任务a6会与分支一、二并行处理

5. 待所有分支处理完成后，流程合并

6. 继续处理串行任务a7



五、未来的展望

最后就是我们对未来流程平台的几点考虑。

智能化

目前告警种类过多，很多流程上的异常告警，在经过人工分析后，多数是不需要处理的，我们的目标是系统能做一些基础的分析，并在自动诊断问题后，对问题进行自我恢复，对于有些需要人工介入的，也能做些初步的分析，同时把经过分析后的日志发送给处理人员，这样可以提高处理人员的效率。

自助式的流程编排

目前的流程的开发还是需要流程团队参与，我们期望在未来，所有的用户可以自行编排流程，自行定义流程中的每一步工艺，所有的工艺可以发布到一个共享的市场，当用户需要编排流程的时候，除了定义自己的工艺外，也可以引用其它团队已经写好的工艺。