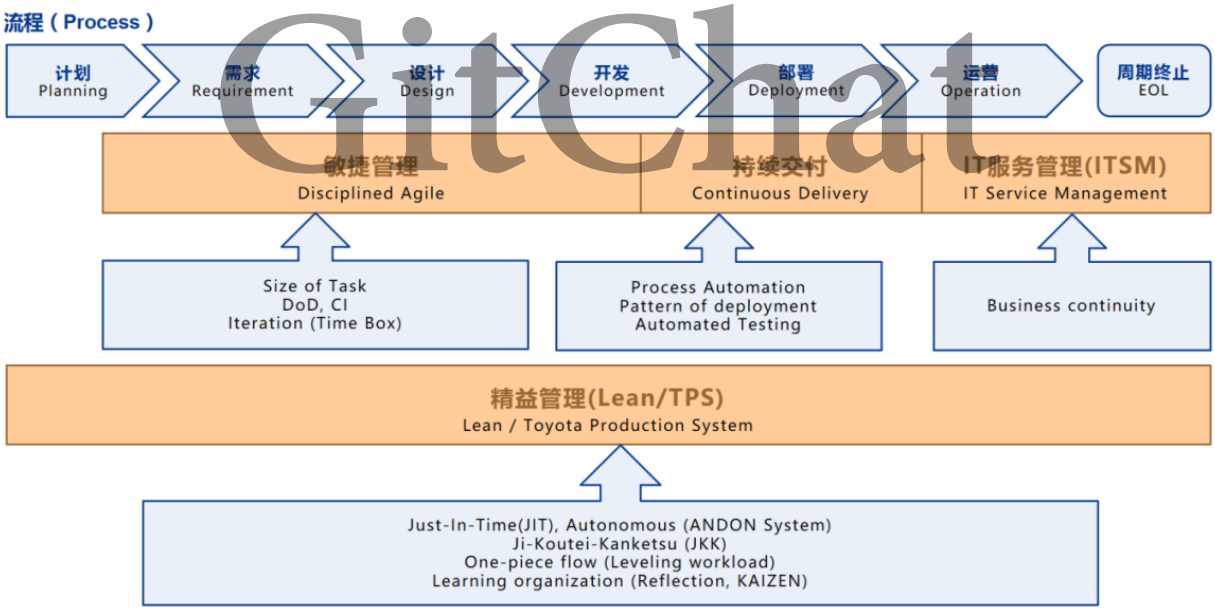
DevOps落地切入点的确定及实施实例

DevOps这个词已经充斥在各个技术论坛，很多企业都说要实行，但真正能落地的却不多。另外很多公司的DevOps只是停留在Ops部门，并不是真正的DevOps。DevOps是贯穿了业务，研发，运维的全过程，所以如何选择切入点就很重要。从目前的很多案例来看，最多的切入点是在Ops，因为运维自动化是有着最成熟的开源工具，同时也是最容易实行的，因为不牵涉到其他部门，关起门来自己玩就好。其次是从测试向两端推进，测试自身也有很多大量可以自动化的工具，同时测试环境的维护也有着Ops相似性。测试反馈的质量问题也可以倒逼开发进行变革。

先介绍一下我所在公司的背景，国内第一家支付公司，有着十几年的历史。从上面大家可以看出什么呢？就是这个公司有着沉重的历史包袱。所以流程很老，思维也很老。对它的改造也会非常的困难。对公司现状进行分析之后呢，发现痛点主要是在几个方面，1，缺乏全局的需求视图。2，开发时间延误，质量低。3，测试效率低。4，上线流程漫长，失败率很高。这几个痛点很多公司也都会有。我们在DevOps的白皮书里，会看到一个完整的流程应该是这样。



那么我们，知道模型之后，我们怎么去尝试呢。等我们开始实行后，如何确定下一步目标呢？就要用到另外一个概念，叫做成熟度模型，最早在软件CMM流程里面，就用了这样一个概念。对于DevOps的持续部署理念也是有这样一张图。

这里面很明确提出，在不同方面，我们的成熟度的不同阶段应该是什么样的？有了这样的一个目标之后呢？切入点如何选择？史记-货殖列传有一句话：天下熙熙，皆为利来；天下攘攘，皆为利往。我们推行DevOps不是为了赶时髦，而是为了利益。所以要找到符合下面几点的切入点：

- 能最快改变现状。
- 能最快见到收益。
- 能最容易用数据说话。
- 能花最少人力。

互联网时代讲求的是效率，天下武功唯快不破。高层往往也是没有耐心的。所以不会给你半年时间慢慢来逐步推行。你需要的是在一个月内能让大家看到改变，看到收益。否则没有人来用你推行的东西就意味着失败。同时要是可度量的，否则有人可以完全抹杀掉你的努力。人力永远是有限的，我的大老板整天说的就是你不要影响我的业务。往往你在进行变革时是在现有人力中来挤出资源做事。反正我的老板不会大笔一挥，给你招几个人来做这件事。所以要规划好人力资源，做一个MVP（Minimum Viable Product最小化可行产品）产品出来，然后继续在上面迭代。最后一点，不要运动式的全面铺开。有些企业是可以发起运动的，例如：华为。但大部分企业文化和员工的接受度不允许你来迅速变革。我在不同公司经历的几次运动式变革都以失败告终。所以要用小刀割肉，割一刀看一下反应，没问题就继续，如果遇到了强力反抗，就要重新评估策略和做法了。

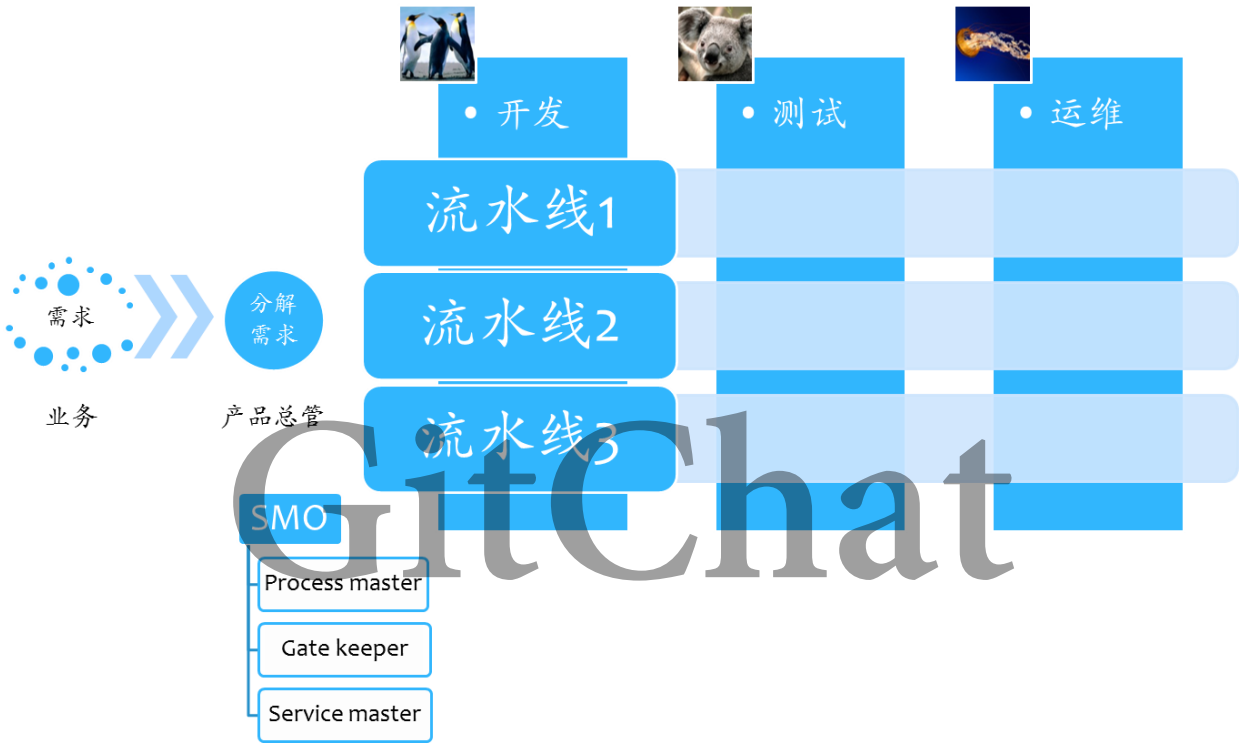
选定好初步的方向后就要继续思考：

- 谁关心痛点
- 谁会从中收益
- 谁会利益受损
- 谁会阻碍
- 谁会支持
- 谁是合作方
- 谁是冲突方
- 谁是旁观者
- 谁来推动

不同的部门之间是有壁垒的。所以当你试图去改变别人的流程、方法时必然会产生冲突。然后你要考虑的是由谁去推动这件事，是你自己，还是同盟者。例如：你是开发工程师，你能做什么呢？可能只能在小团队里推行一下SCRUM或者kanban开发，但测试都不会进入团队，最后这个仅仅是个像SCRUM的半吊子。所以推动者至少应该是中高级管理层。从开发的角度应该拉住测试及运维一起来解决各自的痛点。例如：信息不一致。测试驱动开发。这些都是需要不同部门之间的配合。而这些活动会打破部门壁垒，必然影响到有些人的利益。

例如：有的人权力欲望很强，SCRUM团队把开发，测试，运维绑定在一个团队内就形成了矩阵制结构，直属领导的控制力必然就会被削弱，他指手画脚的机会就少了，所以就

会觉得自己的利益受损。必然会阻碍你推动变革。这也是很多变革失败的原因，触动了太多的利益方。当你看了很多DevOps成功案例后肯定会头脑一热，大喊一声我们干吧。这时候你需要的是冷静的按照上面的几点思考一下，你的同盟者和冲突方的力量对比。很可能发现无法推行下去。所以这也是很多DevOps失败的原因。理想很丰满，现实很残酷。
我是很幸运的，因为研发流程由我管控，测试，配管都在我部门内。运维的管理者和我一起通过的DevOps Master认证。CTO也很支持。所以同盟者的力量比冲突方大很多。在公司内进行了几次分享，描述了美好的愿景后，成功的吸引了大家的注意力。下面就是要考虑如何落地。如果不对现有组织结构进行调整是无法达到，所以经过讨论，按照白皮书里的组织结构设计了这样的一个结构。



组织结构改造完成之后呢，就可以变成一个完整的流水线来进行。不过组织架构调整是个很难的事情，往往由于部门壁垒导致不能闭环。所以要先考虑谁能决定组织架构的调整，现有的流程是否需要大改，现有人员的观念是否容易改变。同时不是所有的开发内容都能变成流水线，因为某些技术限制，没有足够的人员。所以很多时候仍然需要混合新老的组织结构。

很多公司有沉重的业务压力，为了稳定，管理层最看重的是不管你做什么都不能去影响业务。所以比较现实的情况就是我们需要做的事要进行试点，然后以点带面。那么首先从开发的角度来说，你就要选择一个团队来进行开发模式的改变。选择好一个契合度比较高的团队，就是产品开发测试都要能接受这样的变化，然后对他们日常的开发模式，按照DevOps进行改造。另外一点的，DevOps覆盖的流程是很长的，所以也不可能一下全部改变。所以你还是要选择一个切入点。选的这个点要很容易来看到实现效果及收益。可以分析一下不同职位上能获得的收益：

公司的高管：

- 能了解业务需求的进展

- 能看到研发投入

产品：

- 需求更快速被消耗
- 能看到需求的进展

开发：

- 不需要关注和代码无关的事务

测试：

- 更少的重复测试
- 更好的测试覆盖率
- 更好的沟通

配管：

- 更少的重复劳动
- 更少的沟通浪费

运维：

- 更少的重复劳动
- 更少的沟通浪费
- 更快的回滚
- 更少的发布错误

把以上所有的内容都分析好就能确定你的第一个切入点是什么。从研发角度来看可以选择用一个管理工具把信息集中，或者推广SCRUM或kanban开发。从测试角度可以进行代码变动后的自动部署及自动化测试。从配管角度可以选择持续集成。从运维角度可以选择自动生产环境部署和回滚。我选择的切入点是管理工具、SCRUM团队、自动部署。因为原有的Redmine信息不完整，大量信息不对称的情况，计划形同虚设。配管整天手工上线，疲于奔命。开发效率低，延误情况很严重。

在实施过程当中要注意要用利去引诱改变，不要试图立刻改变所有的现状，因为你如果改变所有现状，把所有的东西搞乱，搞乱之后，就会影响业务。影响了业务你就有了大麻烦。所有的改变必须是局部的，影响范围是可控的。

要把资源向敏捷团队倾斜，要给他们提供便利。因为变革前期必然会混乱和低效率，如果不去注意改善，就会变成坏榜样。如果变革使得他们的工作更顺利，这样的就树立了

一个典型，使得其他团队可以看到这么做是有好处的，然后在适当的时候你就可以一刀切，全面推行。下面给出一个例子，我把上线的窗口定义成：

1. 项目上线日期为工作日的周一-周四，自动上线8点45分开始工作，处理前一日完成测试的需求。

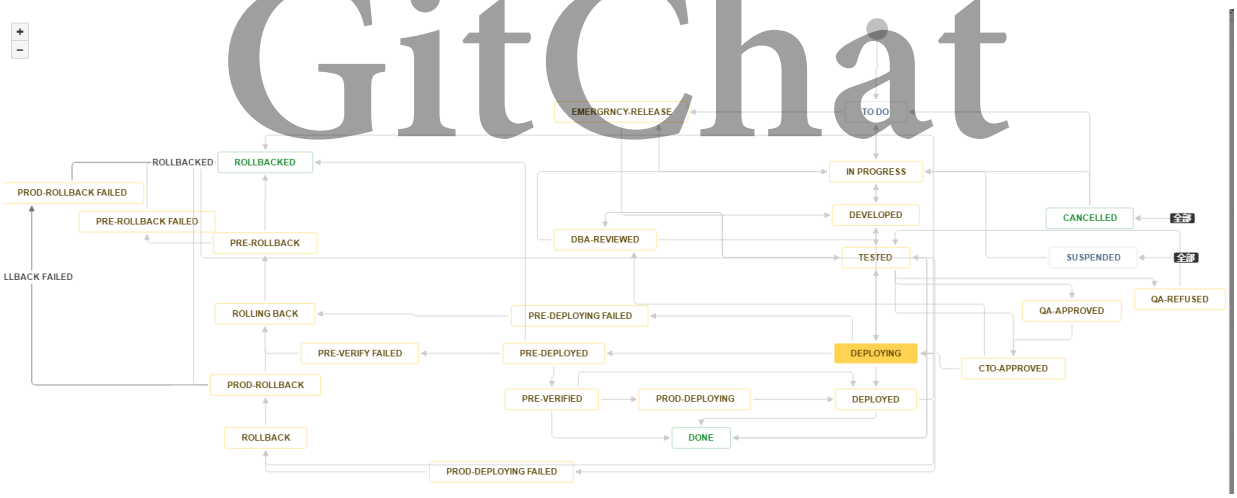
2. 敏捷项目上线日期为所有工作日，自动上线11点-17点处理测试完成的需求。

3. 手工上线的需求上线日期仍然为周二到周四。

这样就吸引大家进行迁移，有些项目会主动选择敏捷开发模式。

下面说一下工具选型的过程。流程软件放弃了原有的Redmine。从DevSuite，禅道，Jira中选了Jira。DevSuite适合超级大公司，看重管理审核。禅道过于简单，和其他软件集成能力弱。Jira有非常丰富的插件库，同时有非常成熟的开发接口和开发库。同时在世界范围Jira有几千家公司在用。所以最终选择了Jira。测试插件用了Kanoah Tests和JIRA Capture。版本控制在单纯git和github、gitlab中选择了gitlab。主要是比较认可gitlab flow。CI工具用Jenkins，这个基本没啥可挑的。而且Jira、gitlab、Jenkins都有插件可以互相连接起来。自动化测试使用的自己开发+SOAPUI+APPINUM等。自动化部署是自己开发。

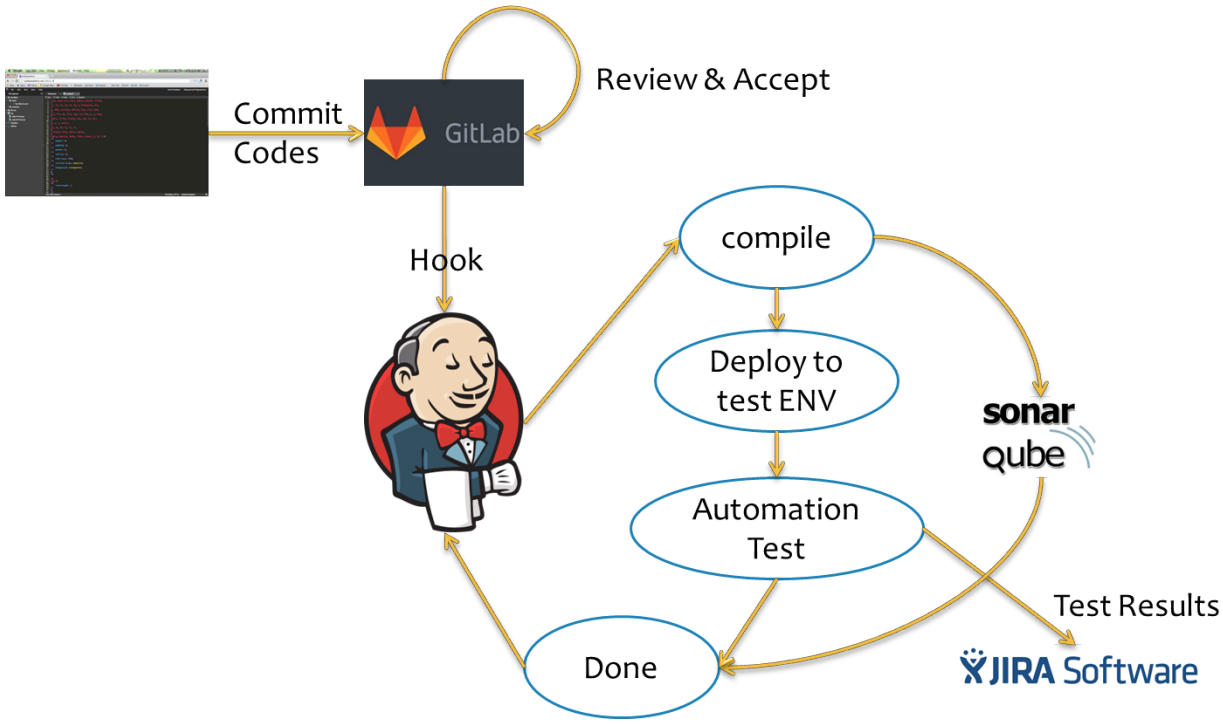
下面是Jira中的一个自动上线需求的状态变迁图：



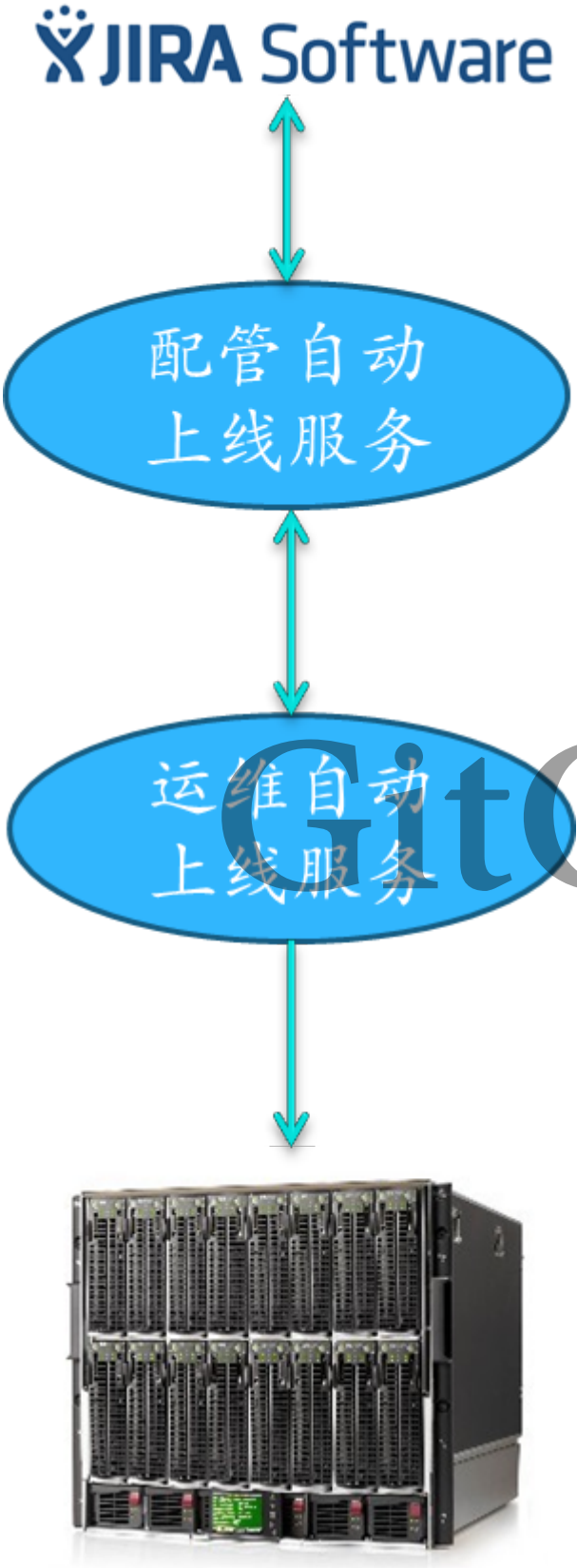
大部分状态都是程序在流转，开发只需要处理2个状态。测试需要处理6个状态。程序使用了Jira的Restful接口来进行操作。

代码的分支使用标准的Gitlab Flow。代码单向流动。MASTER给开发用。PRE-PRODUCTION和集测环境的JENKINS集成，代码变动后自动编译部署及自动化测试。PRODUCTION分支和系统测试环境JENKINS集成，上线也从这个JENKINS直接拉最终的上线包。

一个完整的持续集成过程如下图：



由于测试网络和生产网络是隔离的。所以自动部署分成两个部分，一端在测试环境属于配管，一端在生产环境属于运维。需求测试完成后，自动上线程序就会按照上线窗口来选择对应的需求进行上线。



在全部系统投入运行后，还有很多事情要持续进行。首先最难的是观念的转变，新流程经过多次培训，文档也已提供，还是有很多人两耳不闻窗外事，继续按照老一套进行。同时敏捷的方式也不是所有人能接受，思维的顽固性影响深远。其次迁移工作的工作量巨大，要逐步对老系统按计划迁移。迁移包括代码从SVN到GITLAB。Jenkins上已有脚本

的迁移。测试环境的重整。Jira上流程也需要逐步完善，例如：后期我们把申请新发布单元、紧急上线审批、线上数据修复等流程也放入Jira。再增加各种数据分析和报表。最后一点就是定期对照成熟度模型，看可以进行哪方面的改进。

总结一下最关键的几条原则：

- 要根据自己所在的位置来选择变革的范围。
- 要分析利益相关方来确定变革的阻力。
- 要根据现状和条件来选择最容易突破的地方。
- 要通过利益引导来吸引大家主动改变，同时在适当的时候一刀切。

以上就是我自己的方法论和一些实践经验，欢迎大家讨论。