中国DevOps社区

守望者

敏捷漂流记

第五期工程实践九大坑

出品：中国DevOps社区
作者：陈文峰 王英伟 李岩 方正 黄鹏飞 王东喆 刘志超 刘陈真 胡帅
排版：成芳

卷首：《工程实践之九大神坑》

分支管理没规范，集成发布真混乱。
度量指标瞎乱选，虚假繁荣吹上天。
结对编程来牵线，基情四射笑开颜。
测试全靠人肉点，加班加点累瞎眼。
覆盖不全无人管，出了问题挨个怨。
自动测试年年喊，一到落地就犯难。
蛮力集成就是干，部署挂了都滚蛋。
安全问题讨人厌，频繁被黑真是惨。
技术债务天天欠，不思偿还就是懒。

李岩
项目管理专家&敏捷教练

第1章 洪七公：分支模型不统一，何谈高效交付

九阴真经》产品完成了几个版本上线后，郭靖又接手了《九阳神功》产品。《九阳神功》更为复杂，随着核心业务故事情节逐渐展开，郭靖管理协调的研发团队也越来越多。但郭靖最近在晨会中发现，多个用户故事出现延期，原因都是因为兄弟团队没配合好。产品版本迟迟不能发布，研发团队间的冲突却越来越大，这让郭靖的眉头拧成了疙瘩，一筹莫展。

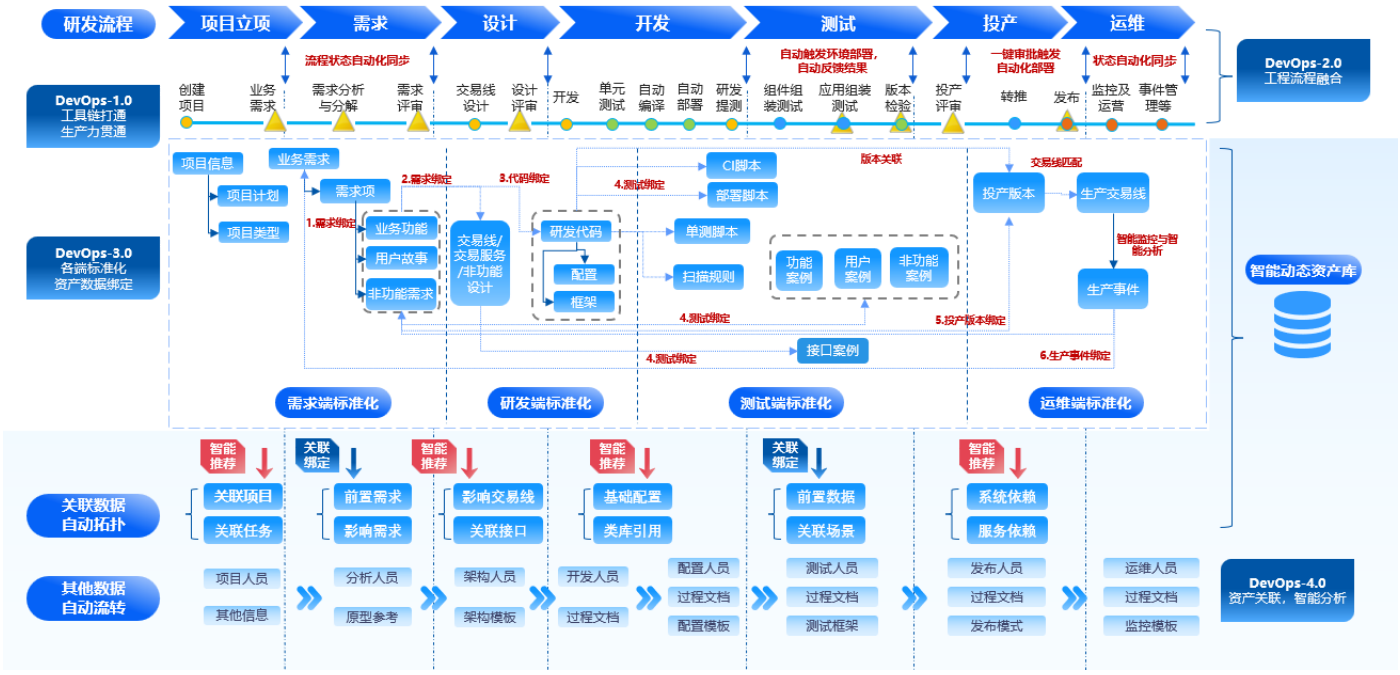
这一天，郭靖又躲在房间里深思，突然听到外面丐帮弟子来报，老帮主洪七公来到。郭靖赶紧出来迎接，师徒见礼完毕，郭靖知七公喜欢吃鸡，带七公到著名的烧鸡店“仙庙烧鸡”。七公一见烧鸡，眉开眼笑，立即大快朵颐起来，一口气干掉两只烧鸡，连连称赞，抬头一看，却发现郭靖一口未吃，忙问：“靖儿，怎么不吃呢？是不是有什么心事？”

郭靖一听，连忙一五一十的把目前团队遇到的问题和困难向七公和盘托出，末了郭靖说：“前两个版本产品都是正常进展，需求也有澄清、开发也有进行技术方案设计，但不知怎么最近就是连连延期，不单是发版本，连转测的DOR都难达到”。

七公拿个大鸡腿，边啃边思考，啃完又仰头喝一大口酒，问：“靖儿，你们这几个团队的研发工程流是怎样的？”

郭靖问：“什么是研发工程流？不就是需求-开发-测试-发布吗？”

七公伸手一掏，掏出一本发黄的秘笈递给郭靖：“你刚说的是需求价值流动，但在底下需要有规范的研发工程流支撑，否则别提需求价值正常交付，连转测都难。”

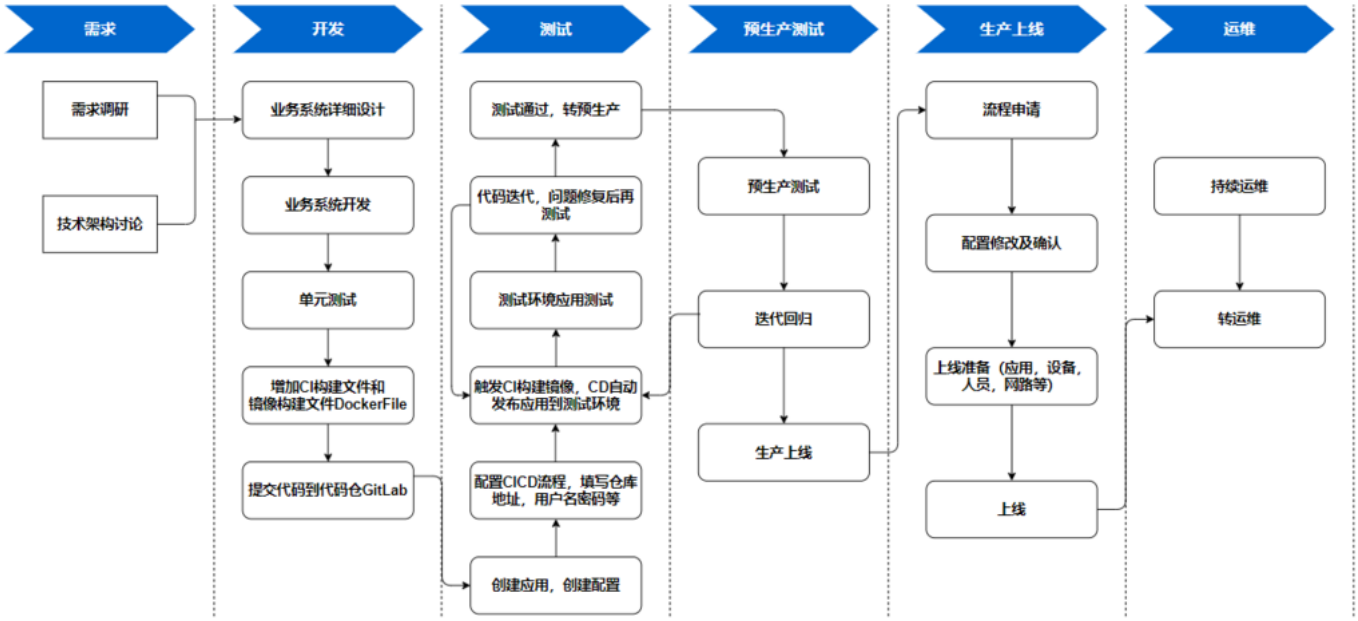


郭靖一听连连点头，接过秘笈一看，封面上书《丐帮软件研发工程四式》（文中简称《研发工程四式》），不禁喜出望外，连连陪七公喝了几杯。

酒席完毕，郭靖回到住处，从怀里掏出《研发工程四式》细细查看，郭靖的思绪也不禁沉浸到这套让丐帮发展壮大的招式当中。话说当年，江湖纷争，群雄并起，洪七公领导的丐帮在江南一带开拓软件产品业务，其它门派也不甘示弱，纷纷涉足其中。洪七公当年敏锐的发现，只有扎实的研发工程支撑，才能保证业务高质快速的交付，于是组织丐帮8大长老，梳理总结，最终整理《研发工程四式》，从而让丐帮软件持续交付神技名扬天下。

第一式 研发流程标准化

虽然经过《九阴真经》产品历练，原来产品团队对Story的DOD已经明确，但怎么能达到DOD各个团队却有不同的套路。怎么整合团队，提升整体效能致力于价值交付，万法之根本在于一致性。保障一致性的基础，在于定义一套标准化的研发流程，统一大家对研发环节的认知，如图一研发流程中清晰定义整个需求生命周期，梳理了需求的研发流程标准环节。借助统一的研发流程定义，拉通对齐各团队对研发流程认知与节奏，建立协同研发的基础。



（图一 研发流程）

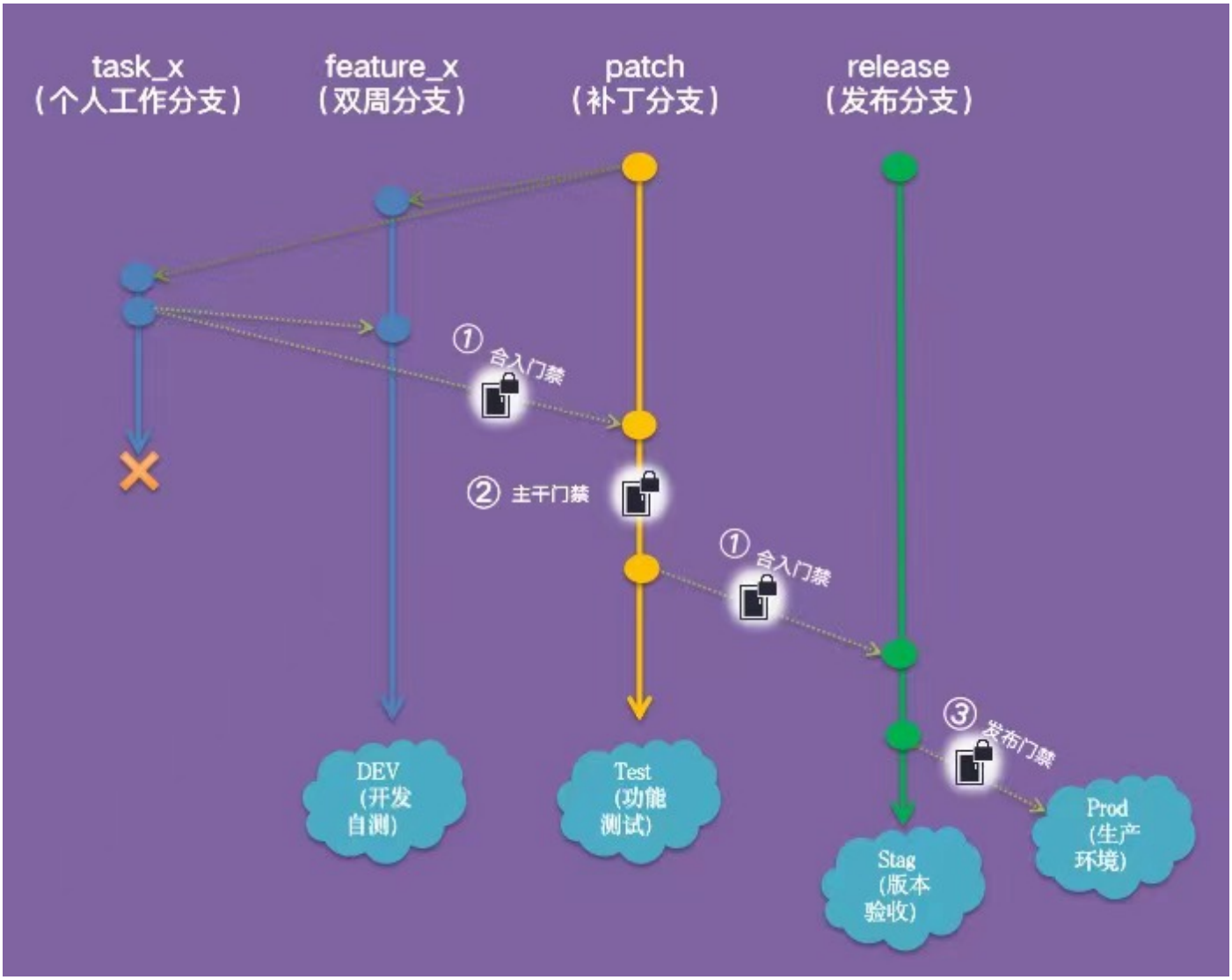
第二式 分支模型统一

郭靖看完第一式，不禁连连称绝，但突然想起来前两天晨会，某个功能两个团队联测的情况，两个团队都说开发完了，但测试却反馈功能无法测试，这又是为何呢？郭靖接着翻开心法第二式，分支模型统一。

分支模型(branching model)，即一个围绕项目（开发、部署、测试）等工作流程的分支操作（创建、合并等）的规范集合。

虽然研发流程定义清晰，但如果分支模型不统一，各团队按自己习惯操作，往往会造成代码合入遗漏或多合，影响测试开始甚至上线版本质量，导致生产事故。最佳办法是在一个组织内统一定义一套或有限套的分支模型，并在组织内部达成共识使用，从而确保在研发过程中不致发生代码、部署问题而阻塞研发。

如图二，在组织内需要清晰定义分支模型，包括分支定义、分支代码合入条件（如门禁、代码审核、功能测试等）、各分支对应的环境。通过定义分支模型从而揭开研发黑盒，统一开发内部代码流向及准入要求，将质量内建至研发过程中。通过管控过程质量，从而确保最终交付质量。



（图二分支模型）

还需要强调的是，随着业务复杂化发展，分支模型配套的测试环境也必需一致齐备，否则会存在A应用有验收环境而B应用没有，没法完整进行验收情况。故各应用需具备一致的环境，包括应用服务、配置、数据库，且同类环境互通、非同类环境隔离也需要确保，否则也很容易出现集成测试时功能、数据各种异常。

第三式 常见分支模型推荐

高效的持续交付体系，必定需要一个合适的代码分支模型。分支模型有利于规范开发团队，遵循统一的规则执行功能开发、问题修复、分支合并、版本迭代及发布等操作，合适的分支策略可以使团队合作变得平滑顺畅，项目有序向前推进。因此，企业的研发团队通常需要慎重地选择代码分支策略应用。

代码分支从大模型的原则上分为三类，一类是主干开发分支发布，一类是分支开发主干发布，还有一类是主干开发主干发布。这些原则的沉淀，一方面是由于代码管理工具的历史发展所导致，另一个方面也受业务发布的时效诉求和管理诉求所影响。基于Git模式的代码管理已经成为绝对主流，基于Git的常见的代码分支模型有Git flow、Github flow、Gitlab flow、TBD flow等。各分支模型对比如图三分支模型对比，可根据组织情况选择合适的分支模型。

分支模式	pros	cons
TBD	1、分支少，实践简单 2、小步快跑 3、合并冲突少 4、利于持续部署和持续交付的支持 5、支持单个软件单个版本	1、对团队的协作成熟度和集成纪律有很高的要求 2、需要有非常好的集成验证基础设施和手段 3、并行开发工作的隔离必须有软件设计和实现的支持
Git-Flow	1、支持特性并行开发 2、规则完善，分支职责明确 3、支持复杂大团队协同开发 4、利于传统软件的发布 5、支持单个软件同时多个版本	1、分支数过多，规则比较复杂 2、分支生命周期长，合并冲突比较频繁 3.对发布后，发布版本的管理需要较多的维护工作
GitHub-Flow	1、支持特性并行开发 2、有明确的协作规则定义，规则简单 3、持续集成，持续部署 4、利于持续部署和持续交付的支持 5、利手与 GitHub 的功能进行集成 6.、支持单个软件单个版本	1、对团队集成纪律有较高的要求 2、对集成和发布的分支管理没有定义 3、集成中，一次中断，影响所有，需要有非常好的集成验证其础设计和手段支持 4、有开发分支周期过长，合并冲突的潜在问题
GitLab-Flow	1、支持特性并行开发 2、有明确的开发分支和发布分支的规则定义 3、利于与 GitLab 的功能进行集成 4、master，pre-production，production 滚动验证发布的方式，支持持续部署和持续交付 5、分支发布方式支持传统的软件的发布方式 6、支持单个复杂软件单个版本及单个软件多个版本两种模式	1、开发分支生命周期过长，合并冲突潜在问题 2、发布中分支与环境之间耦合(pre-production 和 production 这样的分支与环境之间的耦合〕 3、想同时适用多种发布模式的分支方式，同样引入了更多的复杂性，目的不单一

（图三 分支模型比较）

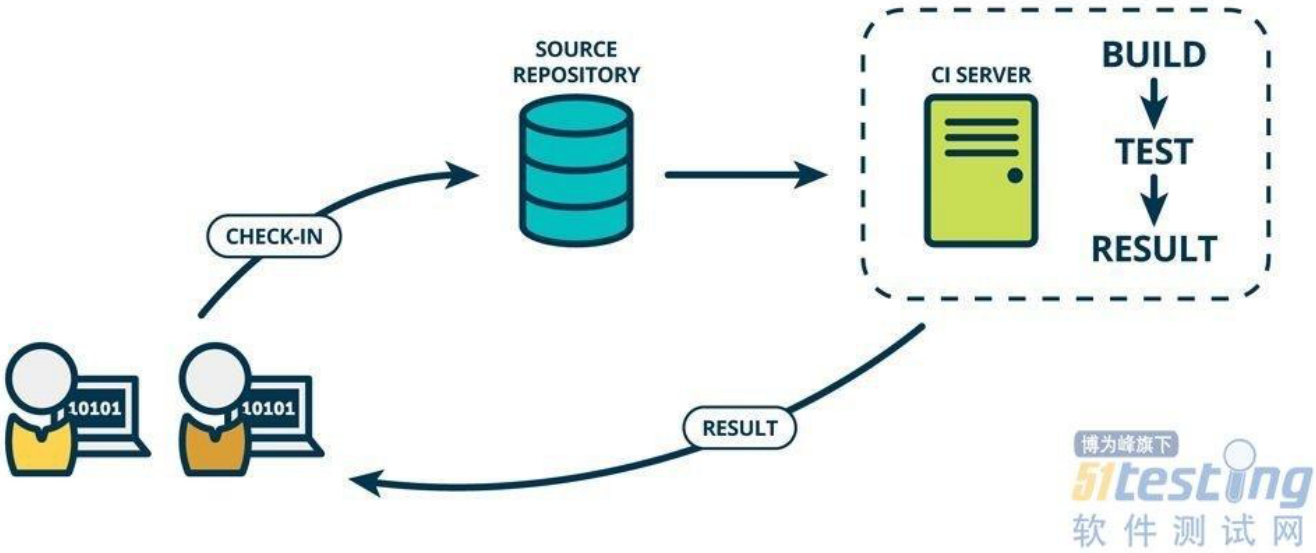
第四式 持续集成

研发流程标准化和分支模型统一后，修炼稍有小成，但需要发挥最大威力，还得炼就最后一式：持续集成。

武学大师Martin Fowler对持续集成是这样定义的：持续集成是一种软件开发实践，即团队开发成员经常集成他们的工作，通常每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动化测试)来验证，从而尽快地发现集成错误。许多团队发现这个过程可以大大减少集成的问题，让团队能够更快的开发内聚的软件。

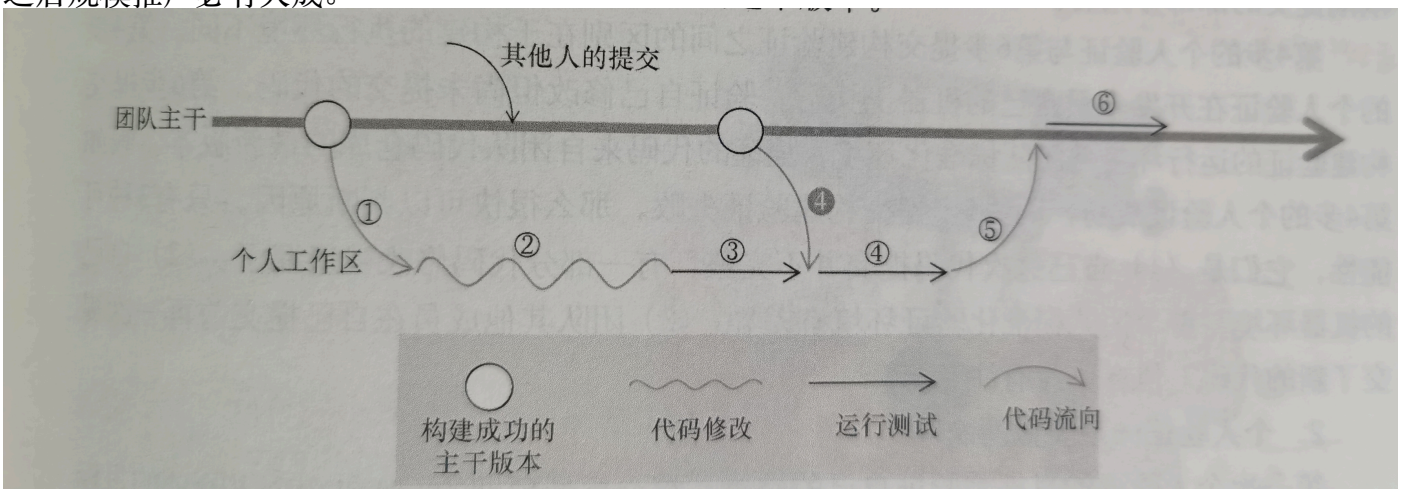
持续集成的宗旨是避免集成问题，如同在极限编程(XP)方法学中描述的集成地狱。持续集成并非普遍所认为是用来改善集成频率的方法，而是一种质量反馈机制。通常的集成过程如图四持续集成所

示，开发提交代码到远程仓库，持续集成服务器自动构建、测试、反馈结果，开发及时调整优化。



（图四 持续集成过程）

在涉及团队成员众多时，需要细化约定代码提交步骤，江湖上流传已久的是本帮乔帮主所创“六步提交法”，步骤如图五六步提交法，可在团队中规模使用推广，更可结合固化成持续集成流水线，久练之后规模推广必有大成。



（图五 六步提交法）

集成流水线举例

流水线类型		预构建	集成构建	版本验证流水线	发布流水线
目的		合入SIT分支前 检查代码质量	合入SIT分支后 检查代码质量	验证版本质量	部署生产
代码分支		Feature分支	SIT分支	UAT分支	Release分支
触发条件		创建PR时自动触发	PR合入SIT后自动触发	何如UAT时主动触发	上线发布前，自动触发
执行任务		代码扫描 单元测试	代码扫描 单元测试 编译打包 部署Dev环境 冒烟测试	打SIT级别版本包 部署SIT环境 自动化回归测试 制品晋级UAT级别 部署UAT环境 业务人员验收 制品晋级RC级别	制品晋级PROD级别 部署PROD环境 冒烟测试
质量门禁	代码扫描	无新增漏洞/缺陷/代码坏味道	无新增漏洞/缺陷/代码坏味道		
	单元测试	通过率=100% 代码覆盖率>80%	通过率=100% 代码覆盖率>80%		通过率=100% 代码覆盖率>80%
	冒烟测试		通过率100%		通过率100%
	功能测试			通过率=100% 接口覆盖率=100%	通过率=100% 接口覆盖率=100%

（图七 集成流水线举例）

郭靖看完心法部份，忍不住喜上眉梢，赶紧拉着黄蓉一起讨论。郭靖未待黄蓉看完就着急问：“蓉儿，你看我们要怎么办？”

黄蓉仔细详读一遍：“靖哥哥，我们可以按照上面所说一步步来修炼，先召集团队核心长老讨论细化研发流程及分支模型，我看咱们底子薄、职责也不清，初期可先使用Git Flow厘清团队职责和规范步骤。”

郭靖连连称是：“还是蓉儿有办法，我马上召集长老们讨论”，郭靖边说边朝门外走去，召集长老讨论确定研发流程及分支模型，明确了统一的持续集成规则，也搭建了持续集成的服务。

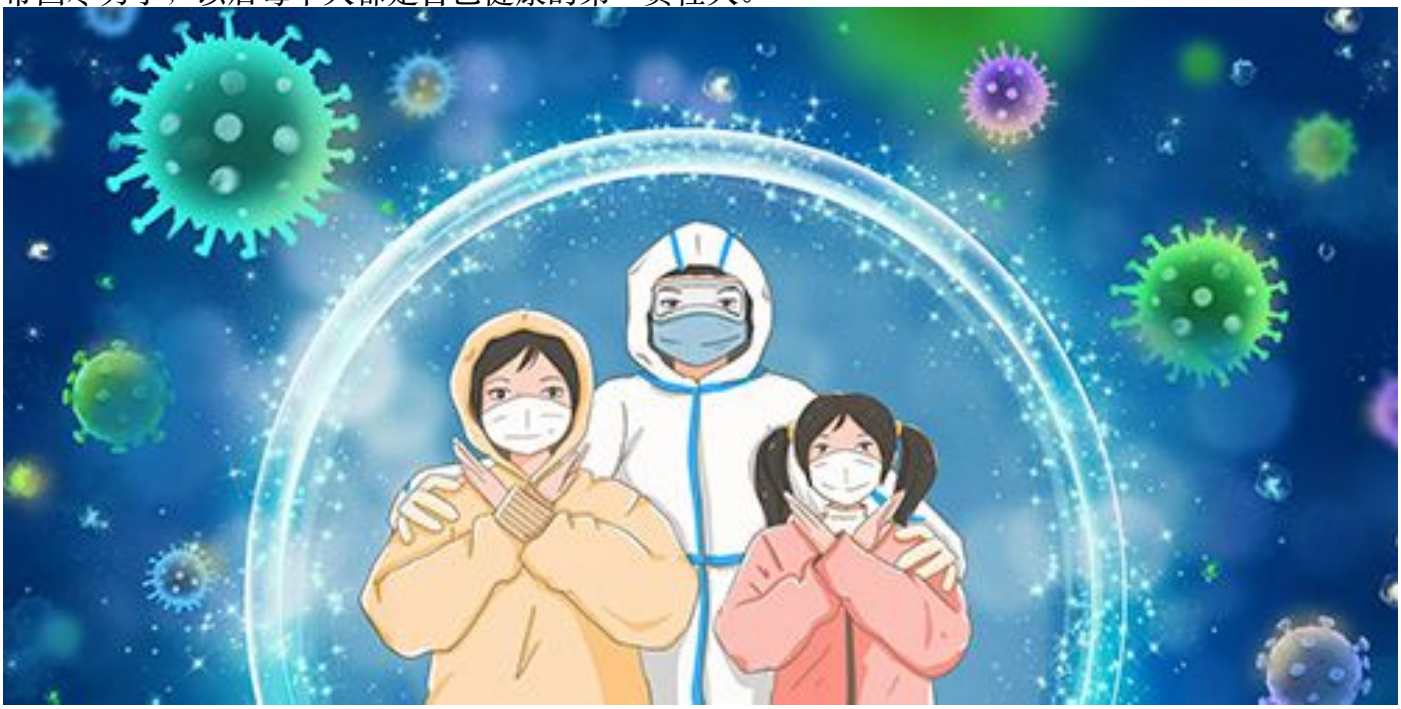
经过一段时间宣讲、对齐和拉通，团队熟悉了这套模式，《九阳神功》研发也终于走上正轨，多个迭代版本也正常发布，团队幸福感也大大提升。

陈文峰
资深研发效能、项目管理者
《运维困境与DevOps破解之道》、《DevOps 悖论》译者

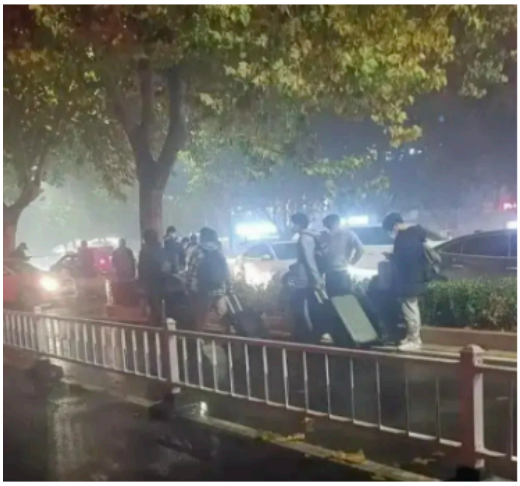
第2章 郭靖：做好数据度量项，效能提升没商量

华夏历开元3022年12月8日，帝都城驿站取消了进京检测24时辰核酸检测的管理规定，标志着华夏帝国的抗疫活动到了一个新的阶段。经过三年的“清零”过程，虽然挽救了无数华夏子民（尤其是老弱妇孺）的生命，但是华夏帝国的经济也受到了前所未有的挑战。

帝国子民三年没有感染新冠，不是因为幸运，也不是因为免疫力强，是因为这三年来每一个接近你的人，帝国都帮你筛查过h了，都是健康的，低风险的，只要有风险的，帝国都把他和你隔开了，帝国尽力了，以后每个人都是自己健康的第一责任人。



自从政策放开以后，全国各地开始了一波感染潮：先是赵子龙的故乡-常山，出门的人比原来少了许多，当地书院的学子，也纷纷返乡。其次，商朝故都-商城，“千里传音器“代工点引起了感染，福克斯康（Foxconn）的学徒也都纷纷归家...



在这个时候，《九阴真经》研发团队也面临着前所未有的困难，帝都研发部门的员工纷纷感染，无法到丐帮信息科技部大楼办公，魔都的测试部门，自动化测试案例修改更新也不及时，导致测试案例无法正常通过，南粤城基础设施团队，提供的持续集成流水线模板，也不符合研发团队的个性化要求。年底奖金发放就在眼前，如果不想出解决办法，会对明年的工作产生巨大的影响。

郭靖作为丐帮CEO，面对这些问题，一直眉头紧锁，找不到解决方案。虽然项目组以前对于各个部门也有过一些考核的指标，但是效果却不理想，比如：

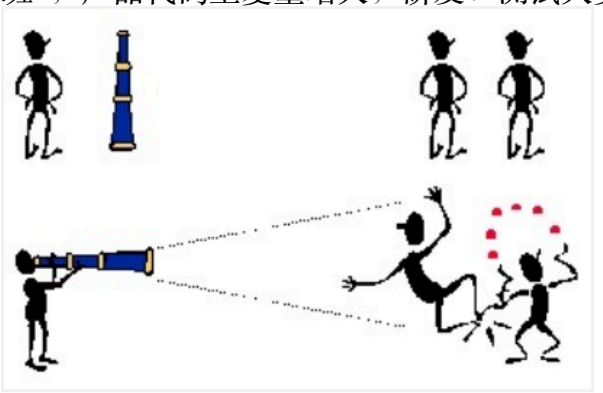
1、产品部门：PO提出的需求数目、点击量等

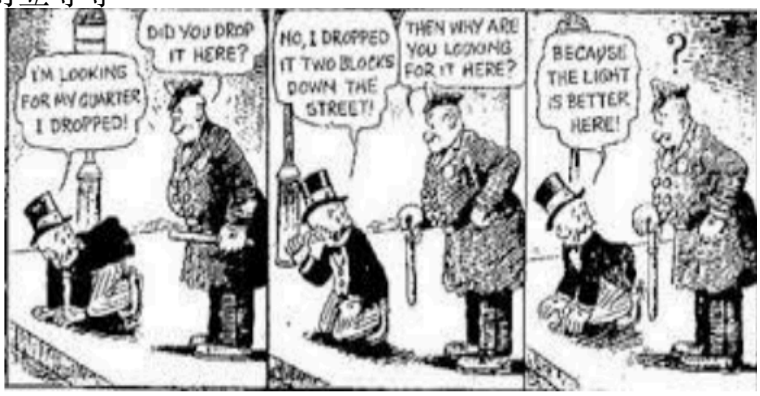
2、研发部门：开发人员提交的代码行数、功能的上线时间等

3、测试部门：测试工程师提出的bug数量，产品功能的测试通过时间等

4、运维部门：运维工程师解决的告警数量，在线支持的时间等

虽然考核指标都有了，根据指标来看，团队的执行效果都还不错，但是对于产品、对于部门的整体效能来看，却没有显著的提升，有些部门反而起到了负面的作用。比如导致部分人员开始“表演型加班”，产品代码重复量增大，研发、测试人员对立等等



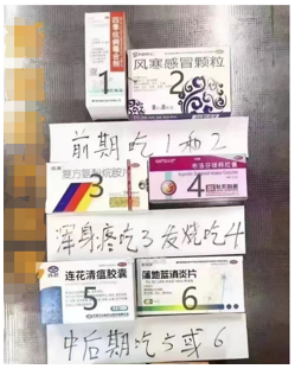


这日，郭靖的外孙（郭芙之子-耶律思杨）也感染了疫症，他去药房抓药。在抓药过程中，他看到了这张图。

阳过

阳康

王重阳



新冠奥密克戎“小阳人”参考症状

概述：症状不像感冒，倒像是来的更猛烈、病程更短的发烧。传播性很强，但感染后靠自身免疫力完全可以痊愈。

时间	症状									应对方案	备注
	发烧	上呼吸道疼痛	浑身酸痛	精神不济失眠	肠胃不适	流鼻涕有鼻音	干咳	味觉	其他		
第1天	身体开始微微发热								核酸阴性	大量饮水	建议同时服用连花清瘟胶囊等抗病毒药物
第2天	开始发烧37度多，直至38.5度以上	上呼吸道疼痛有种要渴死的感觉	高烧导致浑身酸痛	当夜几乎无眠				味觉减退	核酸转阳	布洛芬+大量饮水（电解质水最佳）	
第3天	今日体温反复，夜间体温降到38度以下	上呼吸道继续疼痛	浑身继续难受	半睡半醒、精神疲乏	肠胃不适，下腹部隐隐作痛			味觉减退	核酸阳性	布洛芬+大量饮水（电解质水最佳）	
第4天	今日体温反复，最终降到了37度以下	症状消失	开始好转	精神开始恢复	肠胃继续不适，下腹部微微作痛	开始流鼻涕	开始干咳，咳嗽时整个胸腔跟着疼	味觉减退	核酸阳性	布洛芬+大量饮水（电解质水最佳）	
第5天	今日体温完全正常	症状消失	开始好转	精神继续恢复	腹部疼痛感消失	流鼻涕症状继续	干咳、声音嘶哑、轻微头痛	味觉减退	核酸阳性	布洛芬+大量饮水（电解质水最佳）	
第6天	今日体温完全正常	症状消失	症状消失	精神完全恢复	腹部疼痛感消失	流鼻涕症状减弱	继续干咳，声音嘶哑	味觉开始恢复	核酸转弱阳	大量饮水（电解质水最佳）	
第7天	今日体温完全正常	症状消失	症状消失	精神完全恢复	腹部疼痛感消失	流鼻涕症状继续减弱	咳嗽有好转，但仍旧有鼻音	恢复味觉	核酸转阴	大量饮水（电解质水最佳）	
第8天						轻微流鼻涕	有一点点咳嗽	恢复味觉	核酸阴性		

细看之后，郭靖不得不佩服药店掌柜的学识渊博。小店铺藏着大智慧，这不就是对症下药吗？根据患者所处的不同阶段，不同病症，采用不同的药物进行治疗。反思过后，他觉得《九阴真经》项目组的度量指标也应该按照思路来进行设计。

1、度量指标设计需要针对团队问题

2、度量指标面向组织、团队进行考核

3、构建研发过程不同阶段产物的研发数据链

第一、不想当将军的是士兵，不是好士兵；不能反应团队问题的的度量的指标，不是有用的指标。

不同的团队，面临的问题多种多样，不同的组织，存在的障碍不一而同。只有通过仔细认真的调研，详尽掌握企业不同层级面临的问题，并能够透过现象看本质，设计合适的度量模型，从根源上暴漏问题，并给出合适的解决方案，才能真正起到度量的作用。

第二、组织存在的问题，不单是某一环节的问题，组织研发效能的提升，也是整个交付过程整体流程的优化。

软件研发过程包含：业务分析、项目管理、开发、测试、运维、运营等各个环节，每个环节存在的问题，都有可能影响产品功能最终发布上线的速度。所以作为反应整个研发过程的度量指标，数据应该涵盖研发过程的各个环节，及时反馈瓶颈点，并可以根据历史数据，给出优化建议。

第三、好的数据度量指标，不仅能反应组织存在的问题，分析问题所在，还能基于不同阶段的元数据，给出优化建议，并能对未来的发展局势，做出预测。

对于度量数据，数据采集的实时性、准确性，一直是指标能否反馈实际问题的关键所在。如果数据是基于人工手动采集，那么其真实性、实时性都大打折扣。只能通过工具对数据进行自动采集，并通过大数据技术，对数据进行宏观分析，才能做到底层的数据贯通，真正构建研发数据链，客观、公正的反馈问题，提升研发效能。

于是，郭靖邀请“华夏帝国DevOps社区“的王东喆、巩敏杰、刘陈真、张思琪等多位专家，和《九阴真经》项目团队一起，讨论构建了一套适合团队自身的研发效能度量指标，并构建了自己的研发效能平台，后面使团队的整体效率提升了1.5倍。

贯通端到端研发数据链

构建组织级研发效能仪表盘

组件工程师能力能力画像

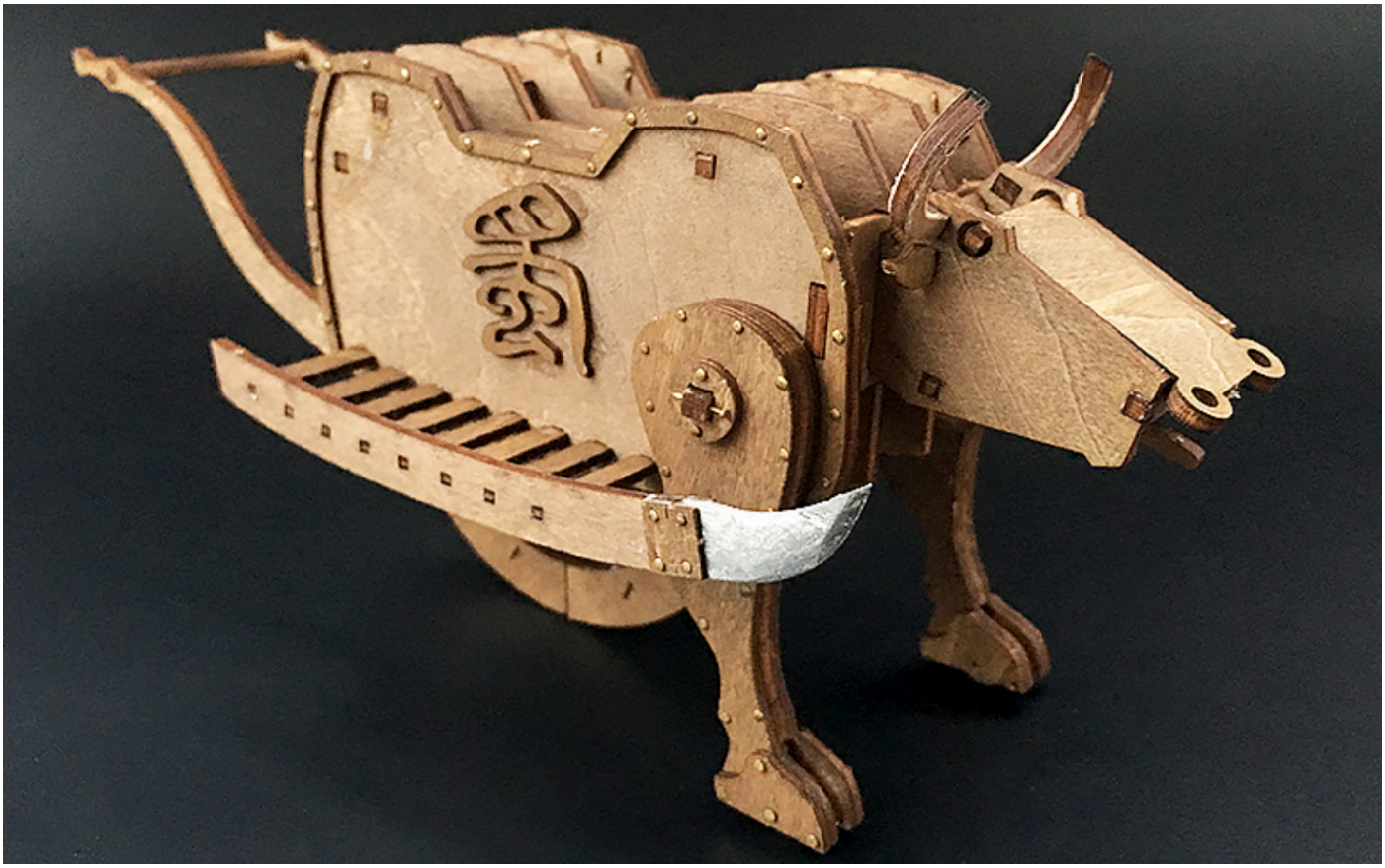
王英伟
DevOps解决方案架构师敏捷/DevOps咨询顾问
软件架构领域最帅的敏捷教练

第3章 孔明：结对编程，可保木牛流马交付无忧

建兴六年（228年）春，蜀汉丞相诸葛亮兵出祁山，开启了克复中原的战争序幕，但因街亭一战失利，蜀汉丧失了闪击曹魏的唯一一次机会，此后，诸葛亮被迫陷入与曹魏的拉锯战中。建兴十二年春二月，孔明入朝拜后主刘禅请求再出祁山，并于昭烈之庙涕泣拜告曰：“臣亮五出祁山，未得寸土，负罪非轻！今臣复统全师，再出祁山，誓竭力尽心，剿灭汉贼，恢复中原，鞠躬尽瘁，死而后已！”祭毕，拜辞后主，星夜至汉中，聚集诸将，商议六出祁山。

临行前，诸葛亮总结了前几次北伐失利的教训，他觉得准备工作不足是最大的问题。而所有准备工作之中，尤其以粮食的准备最为重要。而老奸巨猾的司马懿，也深知这一点，所以每次对阵都是闭门不出，将战争拖入相持阶段，使得蜀军每每受困于粮草不济，最终不战而退。为了克服蜀汉人力畜力不足、道路崎岖的现实困难，诸葛亮将希望寄于运输工具的改造上。于是，他以第四次北伐中使用过的木牛为基础，做了进一步升级，迭代出了流马。木牛用来在蜀中的山道上运粮，流马则用于在煲水、斜水上运粮。

孔明回到帐中，唤裨将杜睿、胡忠二人，附耳授以密计。令唤集随军匠作一千余人，入葫芦谷中，制造木牛流马应用；又令马岱领五百兵守住谷口，并嘱咐到：“匠作人等，不许放出；外人不许放入。捉司马懿之计，只在此举。切不可走漏消息。”马岱受命而去。



杜睿等二人在谷中监督匠作，依法制造，但期间问题不断，主要有以下3点：

第一、随军匠作能力水平层次不齐，新手居多，交付质量参差不齐

第二、部分核心构件重度依赖几个核心骨干，一旦有异动后果难料

第三、匠作间鲜有沟通，自顾自干，协作不顺，而且频繁有人摸鱼

无计可施的杜睿只能上书诸葛亮，求破解之法。诸葛亮接到奏报，沉思片刻，而后提笔写下一行大字——结对编程，可保木牛流马交付无忧。并命人星夜送往葫芦谷。杜睿看后茅塞顿开，但一旁以胡忠为首的众将士们却不知其意。

胡忠问到：何为“结对编程”？

杜睿答曰：结对编程（Pair Programming）是一种敏捷软件开发的方法，也是极限编程（XP）的核心实践之一，简单理解就是两个程序员并排坐在一台电脑前，面对同一台显示器，他们一起进行需求分析、设计、编码、编写测试用例、执行单元测试和集成测试，甚至一起写文档。

胡忠追问：可否将“结对编程”的具体做法细细告之于吾等？

杜睿笑曰：莫急，莫急，且听我一一道来。

胡忠介绍到，结对编程其实有一种标准玩法和一种变体：

第一种、标准玩法——驾驶员领航员模式

按照极限编程定义，参与结对编程的结对角色通常分为两个角色，分别是“驾驶员”和“领航员”，两者的作用如下：

〇 驾驶员控制鼠标和键盘的使用，聚焦于当前正在写的代码，专注于细节，即负责编码工作。

〇 领航员坐在驾驶员一旁观察和思考，即负责检查错误、提出改进的意见，考虑解决方案。

另外，结对的角色是可以随时互换的。

第二种、变体玩法——乒乓模式

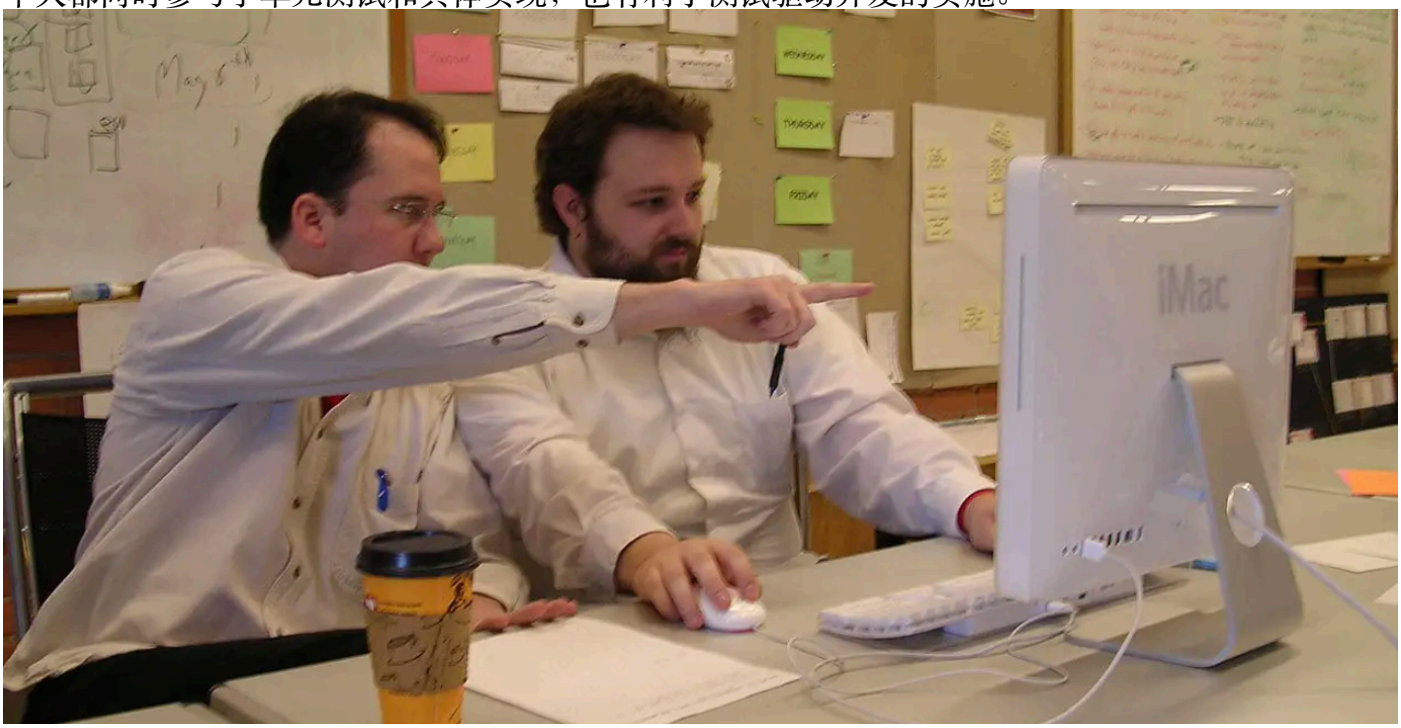
乒乓模式是结对编程和测试驱动开发（TDD）结合的一种变体。具体做法是：

〇 一个程序员先写单元测试，写好单元测试后运行确认不通过。

〇 另一个程序员写具体功能代码，写好后再运行单元测试，确认通过，然后进行可能的代码重构。

〇 写测试的程序员和写功能代码的程序员，角色互换。

以此类推，循环往复。你写单元测试，我来实现具体代码，然后我再写单元测试，你来实现具体代码，就像打乒乓球一样。这样的好处是单元测试和代码实现是不同的人负责，更能避免遗漏，且两个人都同时参与了单元测试和具体实现，也有利于测试驱动开发的实施。



众匠作听后都觉得新奇不已，正在这时，人群中突然冒出一个声音：胡将军，这么做有什么好处啊？话音刚落，大家纷纷附和：对啊，新奇却不知有何用处。

杜睿捋了捋胡须，对众将说到，其利有三，具体如下：

第一、可促进沟通，加速人员成长，以提高团队整体战斗力

通过结对的方式，两个人可以互相分享对业务逻辑及代码实现的理解，促进团队内的沟通。尤其是团队中的新手，可以在这个过程中很好的向其他成员学习，快速融入团队并上手，在短时间内提升个人能力，从而提升团队的整体战斗力。

第二、可共享知识，减少个体依赖，以实现代码集体所有制

采用结对编程的方式，两个人为完成同一个交付目标，相关的知识和经验自然要相互分享，很难藏私。同时也使得每个功能模块都会有至少有两个人了解，使得代码集体所有制成为可能，也减少了某系统或某功能点对个人成员的重度依赖，即使将来有异动（请假、转岗或离职等），整个团队的交付能力也不会受到很大的影响。

第三、可减少摸鱼，加速问题解决，以提高交付质量和效率

由于结对编程是两个人在一起工作，所以每个人都更专注，不容易频繁出现分心和偷懒的情况，加速了价值的流动。同时两个人一起，经验互补，相比一个人，可以前置发现很多潜在问题，大大减少了代码中的缺陷，从而提升交付的质量。

一旁的武将马岱听到后，大笑不止，略带讥讽的问到：说了这么多好处，但原本一个人的工作，现在两个人一起做，这成本不是明显增加了吗?

杜睿听后急忙言到：非也非也！研发成本=开发成本+重构成本+测试成本+改bug成本，结对编程虽然会略微增大最初的编码成本（有研究表明结对编程比单独编程仅仅多消耗15%的时间），但是由于代码质量的提高，可以大大节约重构成本、测试成本和改bug成本，同时兼有加速人员成长，减少对个体的依赖，减少摸鱼，加速问题解决等诸多好处，又何乐而不为呢?

终将听后皆连连称赞，于是都依令开始践行结对编程，果不其然，数天后，军匠团队的整体气氛瞬间提升，大家其乐融融，交付的木牛流马质量也明显提高。数月后，木牛流马皆造完备，宛然如活者一般。上山下岭，各尽其便。众军见之，无不欣喜。孔明令右将军高翔，引一千兵驾着木牛流马，自剑阁直抵祁山大寨，往来搬运粮草，供给蜀兵之用。后人有诗赞曰：剑关险峻驱流马，斜谷崎岖驾木牛。后世若能行此法，输将安得使人愁?

然而天命难违，依靠木牛流马解决了粮草问题的孔明，却在两队阵之际不幸病重，孔明强支病体，令左右扶上小车，出寨遍观各营。自觉秋风吹面，彻骨生寒，乃长叹曰：“再不能临阵讨贼矣！悠悠

苍天，曷此其极！是夜，孔明令人扶出，仰观北斗，口中念咒，以保将星不落，而后入塌嘱托后事，答问间渐无声息，众将近前视之，已薨矣。时建兴十二年秋八月二十三日也，寿五十四岁。后人赞曰：

拨乱扶危主，殷勤受托孤。英才过管乐，妙策胜孙吴。
凛凛出师表，堂堂八阵图。如公全盛德，应叹古今无！

李岩
项目管理专家&敏捷教

第4章 老顽童：手动测试vs自动测试，左右互博

已经冬至了，再过2周就马上要新年元旦了，郭靖的《龙象般若功 2.0》差不多就要完工出山了。这个产品，从年中就开始规划，9月份完成了 1.0 ，本来计划12月完成 2.0 的大版本升级的，但剩下两周，还有很多遗留的缺陷未解决，眼看就要到限定的日期了，要是延迟了发布，就错过了抢占市场的大好机会了。作为 PO 的郭靖着急的不行，可是研发老大金轮法王却说：“大伙加把劲，我们再搞10个通宵，这些问题就通通解决了。哈哈哈，大力出奇迹嘛~，做研发哪有不赶工期加班的，大伙好好干，做好了 2.0，我给大家申请4个月的年终奖”。现场的人连头都不抬一下，因为这样的鸡血，已经打了一个多月了，大家都苦不堪言，但是为了目标，都是咬紧牙关撑着。

郭靖也觉得团队非常疲惫，快要到极限了，但实在是测试出来太多问题了，而且每次修复完又要测试团队全部手工回归一遍，太费时间了。同时，因为是基于 1.0 上开发的，测试回归工作量骤增，有些测试人员都要测得快吐了。郭靖一筹莫展，拉着黄蓉就想出去走走。

“靖哥，测试里面好多工作都是重重复复的，我们是不是可以上自动化测试啊，这样就可以释放很多测试人力了”，黄蓉也着急替郭靖想办法

“哪有那么容易，现在研发时间又紧，自动化测试还要写测试脚本，编写测试用例，需要花很多前期投入的。再说了，现在的团队都是从1.0 开始就手工测试，都熟练的很，突然让大家转成自动化测试，搞不好大家还不愿意呢，甚至速度更慢了”

“但自动化测试真的很有必要啊。你看你们龙象般若 1.0 的时候测试用了1个月搞掂最后的版本测试验收，现在2.0 了，用了一个半月都还没搞完。要是明年 3.0 了，那3个月可能都在测试呢，还想一个季度一个大版本，根本不可能。”

“我也知道啊，但哪些可以转成自动化，哪些不能转呢？我之前也没有做过，现在胡子眉毛一把抓，肯定会打乱团队节奏的。“

两人边走边聊，来到了桃花岛上。

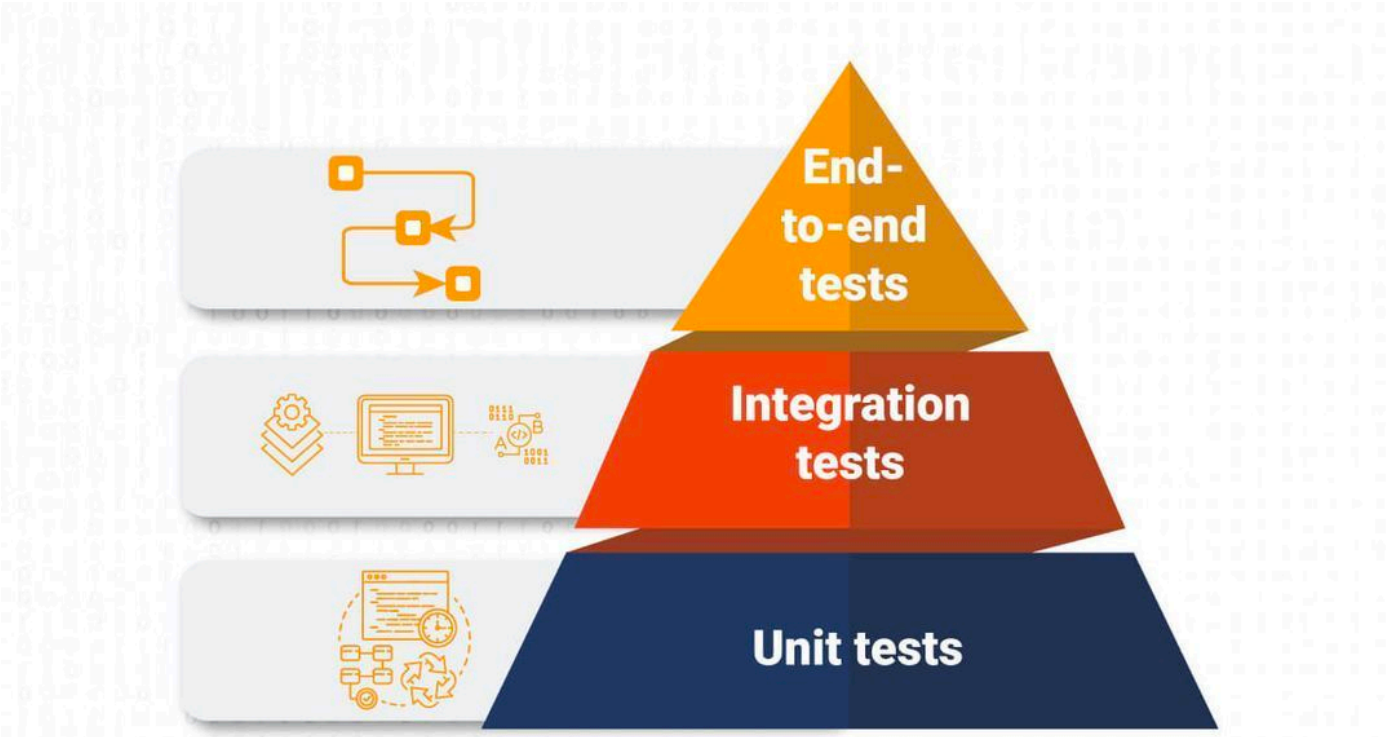
这时，老顽童正值无聊，在自己和自己下象棋，“杀”得有来有往，不亦乐乎，竟然连有人走来也没有察觉。郭靖虽然正愁着版本发布的事，但看到自己的棋友自娱自乐也下得有来有往，便想上去一探究竟。

“自己和自己下棋，下一步要走哪都知道了，有什么好玩的”，黄蓉觉得无聊，先拉郭靖去别的地方，但郭靖执意要去看看棋局。眼看周伯通坐在楚河汉界边上，左手操红棋，右手运绿棋，两边阵势浩荡，一方「守中带攻」，另一方「攻中带守」，完全就是两套打法，真就像是两个不同的人在对垒一样。

美滋滋的看完一局左右手的对垒，郭靖心满意足。他回想起刚才在路上和黄蓉聊的内容，突然有些事情想明白了。“我为什么一直觉得自动化测试就非要替代手工测试呢，他们可以相辅相成的呀！”

郭靖自言自语说道。

老顽童一边收拾着棋盘说道：“是的，自动化测试可以节省人力和时间成本，提高测试效率。但自动化测试并不能完全代替人工测试。自动化测试能解决很多问题，同时也带来很多问题，例如：
第一，在测试覆盖率上，在同等时间内，使用自动化测试能够覆盖更多的功能，但只适合回归测试，在开发中的功能就不划算了。对于开发中的功能，需求的更改，都会导致自动化脚本的变更，维护脚本的工作量和开发这个功能不相上下，维护的成本是很高的。
第二，在测试效率方面，如果每次都是测试相同的数量的内容，那自动化自然是更快的。但开发自动测试脚本要比用例开发耗时长，包括编写脚本、调试脚本、维护脚本，而手工测试也要对测试用例进行撰写、评审、修订。由于用例编写更多为文本动作，时间上肯定会少。
第三，在可靠性上，自动化测试依赖的是脚本，后续定位、复现有明确的配置路径可循。但程序是死的，人是活的。目前而言，最智慧的还是人。可以说是成也萧何败也萧何，自动化的稳定来源于其死板，而人的智慧体现在思维的跳跃，跳跃的思维也会导致后期不易定位。
第四，自动化测试擅长完成压力、负载、并发、重复等人力不易完成的任务，而且只要资源给够，24小时运行都是毫无怨言的。这块就尽快安排转型吧，
最后一点，自动化测试很重要的角色是有测试开发，必须提升测试人员能力，提高与开发沟通的效率。想要培养一名自动化测试人员耗费资源更多，在团队中推广自动化配套的培训、测试管理、产品开发环节都要跟上。



手工测试和自动化测试就像左右手，各有各的好处，要相互辅助，才能最大幅度的提升效率，降低成本。”

郭靖感觉找到了打破僵局的机会了，虽然目前 2.0 可能来不及了，但是后面 3.0 、4.0 要是能改进以自动化测试为主快速回归验证发布，甚至可以在开发环境、测试环境都加上自动化的持续测试，那研发人员就可以更早更快的知道问题所在，不用等集成后或者发版本后才发现问题了。

“我回去就和金轮法王商量，先把大家的单元测试、代码扫描、安全测试都先集成到流水线，一提交就自动测试，然后再把 DEV 环境的集成发布增加一个自动化的接口测试，检查所有集成环境的接口，最后就是每次发版都先用自动化测试用例把回归测试做完，人工测试就只负责新功能测试。”，郭靖把回去怎么改造都想好了

“而且，还可以趁大家晚上睡觉的时候，利用空闲的资源做压力测试、并发测试，还不用担心占用日常的资源。一觉醒来就能看到有什么问题发生，那也算是一个不错的改进了”

郭靖越想越开心，抓了老顽童就说：“来来来，我来和你下一局，自己一个人下多没劲啊～”

方正
中国DevOps社区核心组织者
多年研发、项目管理、产品管理经验

第5章 黎生：人生小人物，测试大追求

武林之中，除了各个武林高手，更多的还是小人物，而在这些小人物之中，就有一位十分值得敬佩的角色，仅仅学习了降龙十八掌之中“神龙摆尾”这一招就成为了人们敬仰的大侠，他就是《射雕英雄传》中的丐帮弟子黎生。他曾经受洪七公赏识，学会了降龙十八掌中的一招，但是却自刎而死。

黎生，丐帮第八代弟子，绰号江东蛇王，黎生生平办事勤奋，所以深受洪七公赏识，给他传授了降龙十八掌中的一招“神龙摆尾”。加上自己也比较勤快，所以进步很快，学有所成之后，经常在江湖里面，行侠仗义，帮助黎明百姓不受欺辱，就像他的名字一样，黎生、黎生，黎明百姓而生，自然要为百姓发声。

这日，正值冬日，新年临近，黎生和三两好友在酒馆喝酒，几杯下肚之后，开始大快朵颐，一边吃肉，一边向好友们说着，初遇洪七公之时的场景，这是他无法抹去的记忆，在遇到洪七公之前，黎生过的一直很困惑，无能无力，做啥都没有目标，直到遇到洪七公，洪七公才帮他答疑解惑，助力他的人生飞黄腾达。

黎生说到：当时见到洪七公的时候，我就问了一个问题，现在我有一个小团队，我们目前主要做一些软件研发工作，但是做出来的软件在质量这块一直比较差，我们自己也想了很多的办法提升，但是见不到效果，洪老您这边有什么锦囊妙计没有，还麻烦洪老指教。这时候洪七公说到：你小子算是问对人喽，当年我年轻的时候，关于这块也踩过不少坑，且听我一一道来，给你一些启示。

首先关于测试，我想说这个对于软件来讲，是很重要的一个动作，这个软件做的好不好用，问题多不多，关键就在于测试，测试的好不好，流程对不对，都对我们的软件质量起了关键性的作用。另外，软件的测试每家公司的情况不一样，不能一概而论，需要脚踏实地的根据实际问题去规划，解决。没有最好，只要合适自己，能解决组织问题，就是最好的。所以呀，我们还是要重视测试的每一步。在这里我也只浅谈一下我对测试的一些想法。我把它提炼为几个字“抓基础、拓手段、跟遗留、善分析”，这个十二字诀送给你，希望可以帮到你。下来我们一起看看，这个十二个字啥意思。

第一步：“抓基础”，所谓抓基础，就是需要我们做好软件的基本测试，比如核心的单元测试。对于程序员来说，如果养成了对自己写的代码进行单元测试的习惯，不但可以写出高质量的代码，而且还能提高编程水平。

单元测试在这里说一下大致：

1、软件中存在的错误发现得越早，则修改和维护的费用就越低，而且难度越小，单元测试是早期抓住这些错误的最好时机。

2、单元测试是对软件基本组成单元（如函数、类的方法等）进行的测试，而且软件单元是在于程序其他部分相隔离的情况下独立测试。

3、一般在代码完成后由开发人员完成，QA（质量管理员）人员辅助。

软件单元测试是软件开发过程中不可或缺的一部分，可以帮助开发团队更好地控制软件质量，确保软件的可靠性和可用性。单元测试可以帮助检查软件功能是否符合预期，检查软件是否存在缺陷，以及检查软件是否满足用户的要求。

总的来说，可以帮助开发团队更好地理解软件的功能。所以我们要先把这个基础打好，打好基础之后，我们就可以去进一步提升测试手段，追求更加高效的测试方法。这就是我们要说的第二句话。

第二步：“拓手段”，所谓拓手段，就是在现有测试的基础上，希望能提升测试的效率，这个效率不仅包含：速度快，更是质量高。所以在这个阶段我们可以借助一些自动化的测试工具，开展更高效的自动化测试。软件自动化测试是一种通过使用脚本和程序来自动执行测试任务的测试技术。它可以帮助软件开发团队更快地完成测试工作，提高质量，减少测试成本。

软件自动化测试的做法有很多，主要有以下几点：

1、首先，要明确自动化测试的目的，确定要测试的功能，以及测试的等级。

2、其次，根据软件的功能，编写自动化测试脚本，编写完成后需要对脚本进行调试，确保脚本能够正确执行。

3、然后，运行自动化测试脚本，收集测试结果，并分析测试结果，检查软件是否满足预期功能。

4、最后，编写测试报告，反馈给开发团队，以及相关的管理人员，以便及时发现问题，改进软件质量。

下来看看自动化测试工具有哪些，常见的自动化测试工具可以分为：UI自动化、性能自动化、接口自动化这些，而随着自动化测试的发展，相信会有更多更好的方法工具出来。

1、Appium：AppUI自动化测试

Appium 是一个移动端自动化测试开源工具，支持iOS 和Android 平台，支持Python、Java 等语言，即同一套Java 或Python 脚本可以同时运行在iOS 和Android平台，Appium 是一个C/S 架构，核心是一个 Web 服务器，它提供了一套 REST 的接口。当收到客户端的连接后，就会监听到命令，然后在移动设备上执行这些命令，最后将执行结果放在 HTTP 响应中返还给客户端。

2、Selenium：WebUI自动化测试

Selenium是一个用于Web应用程序测试的工具。Selenium测试直接运行在浏览器中，就像真正的用户在操作一样。支持的浏览器包括IE(7、8、9)、Mozilla Firefox、Mozilla Suite等。这个工具的主要功能包括：测试与浏览器的兼容性——测试你的应用程序看是否能够很好得工作在不同浏览器和操

作系统之上。测试系统功能--创建回归测试检验软件功能和用户需求。支持自动录制动作和自动生成 .Net、Java、Perl等不同语言的测试脚本。Selenium 是ThoughtWorks专门为Web应用程序编写的一个验收测试工具。其升级版本为Webdriver。

3、Jmeter：接口测试，性能测试

JMeter是Apache组织的开放源代码项目，它是功能和性能测试的工具，100%的用java实现JMeter可以用于测试静态或者动态资源的性能（文件、Servlets、Perl脚本、java对象、数据库和查询、ftp服务器或者其他的资源）。JMeter用于模拟在服务器、网络或者其他对象上附加高负载以测试他们提供服务的受压能力，或者分析他们提供的服务在不同负载条件下的总性能情况。你可以用JMeter提供的图形化界面分析性能指标或者在高负载情况下测试服务器/脚本/对象的行为。

4、Postman：接口测试

Postman 提供功能强大的 Web API 和 HTTP 请求的调试，它能够发送任何类型的HTTP 请求 (GET, POST, PUT, DELETE...)，并且能附带任何数量的参数和 Headers。不仅如此，它还提供测试数据和环境配置数据的导入导出，付费的 Post Cloud 用户还能够创建自己的 Team Library 用来团队协作式的测试，并能够将自己的测试收藏夹和用例数据分享给团队。

5、Monkey：稳定性测试

软件附带在sdk中，适用于android和ios，通过adb shell，生成用户或系统的伪随机事件，压力测试结果：崩溃crash，无响应anr，基本命令：adb shell monkey 300。

6、Robot：WebUI 自动化测试，接口测试

Robot Framework是一款python编写的功能自动化测试框架。具备良好的可扩展性，支持关键字驱动，可以同时测试多种类型的客户端或者接口，可以进行分布式测试执行。主要用于轮次很多的验收测试和验收测试驱动开发。

7、Loadrunner：性能测试

LoadRunner，是一种预测系统行为和性能的负载测试工具。通过以模拟上千万用户实施并发负载及实时性能监测的方式来确认和查找问题，LoadRunner能够对整个企业架构进行测试。企业使用LoadRunner能最大限度地缩短测试时间，优化性能和加速应用系统的发布周期。 LoadRunner可适用于各种体系架构的自动负载测试，能预测系统行为并评估系统性能。

说了这么多，一句话总结：软件自动化测试的意义在于，可以提高软件质量。可以帮助开发者更快地发现软件中的缺陷，从而更快地修复它们，从而减少软件发布时可能出现的问题。

一会功夫洪老就说了这么多，黎生也听的津津有味，洪七公说，让我休息一下吧，小伙子，年纪大了，说多气喘，喝口水缓缓在接着说，黎生听到笑了一下，就马上给洪七公递上水杯。洪七公一边喝水，黎生说，洪师傅啊，这样子打好基础，在有自动化的加持，确实能提高不少效率啊，也能发现不少之前人力发现不了的问题，无论测试的广度还是深度都得到了较大的提升啊。洪七公说道：是啊，但你有没有想过一个问题，就是我们做的这些动作，都是在发现发布前的问题，我们还要更加重视发布之后的问题发现啊，这样子才能形成一个闭环，既有发布前的测试，也有发布后的测试，覆盖了测试的更多边角，不断的追求测试的覆盖性，这样子我们的产品才能用起来更好。黎生听到，连忙点头，说道，洪师傅，这个是不是就要涉及到我们的第三步了。洪七公答道：“对喽，就是第三步，我们接着往下看”。

这个第三步啊就是我们要对发布之后的产品进行线上的测试，不能说，发布了，我们就不管了，这个思想我们要调整，发布了不代表一点问题都没有，我们要把能做的都做完才可以。

软件的线上测试是指在软件已经发布后，对软件进行测试，以确保软件的健壮性和可靠性。这种测试通常是对软件的可用性、安全性、性能、可维护性和可扩展性等方面进行测试。软件线上测试是一种重要的质量保证测试，可以帮助检测出软件可能存在的缺陷，从而避免软件在使用过程中出现问题。这里的测试，每家公司做的都不一样，可以根据自己的实际情况来做。在这里我说说我们怎么做这一块的大致：

1、确定线上测试的轻重点，线上测试很难做到全部的测试，需要有一个测试的清晰目标和范围。

2、明确测试的方法，当大家知道如何开展遗留测试，并且从中受益。

3、线上测试的时机，发布之后什么时候开展遗留测试，做几次等等这些，都需要根据组织情况定义清晰，便于管理和开展工作。

相信有了以上几个方面，大家的线上测试可以做的顺利很多。在做的过程中也能收获不少，感受到线上测试开展的重要性，不断的追求更高的测试，让测试覆盖到我们业务的每个过程，从而提升质量。

黎生听完，直呼：“妙啊”，产品全方面覆盖到了测试，进一步保障了产品质量。你真厉害，洪师傅。

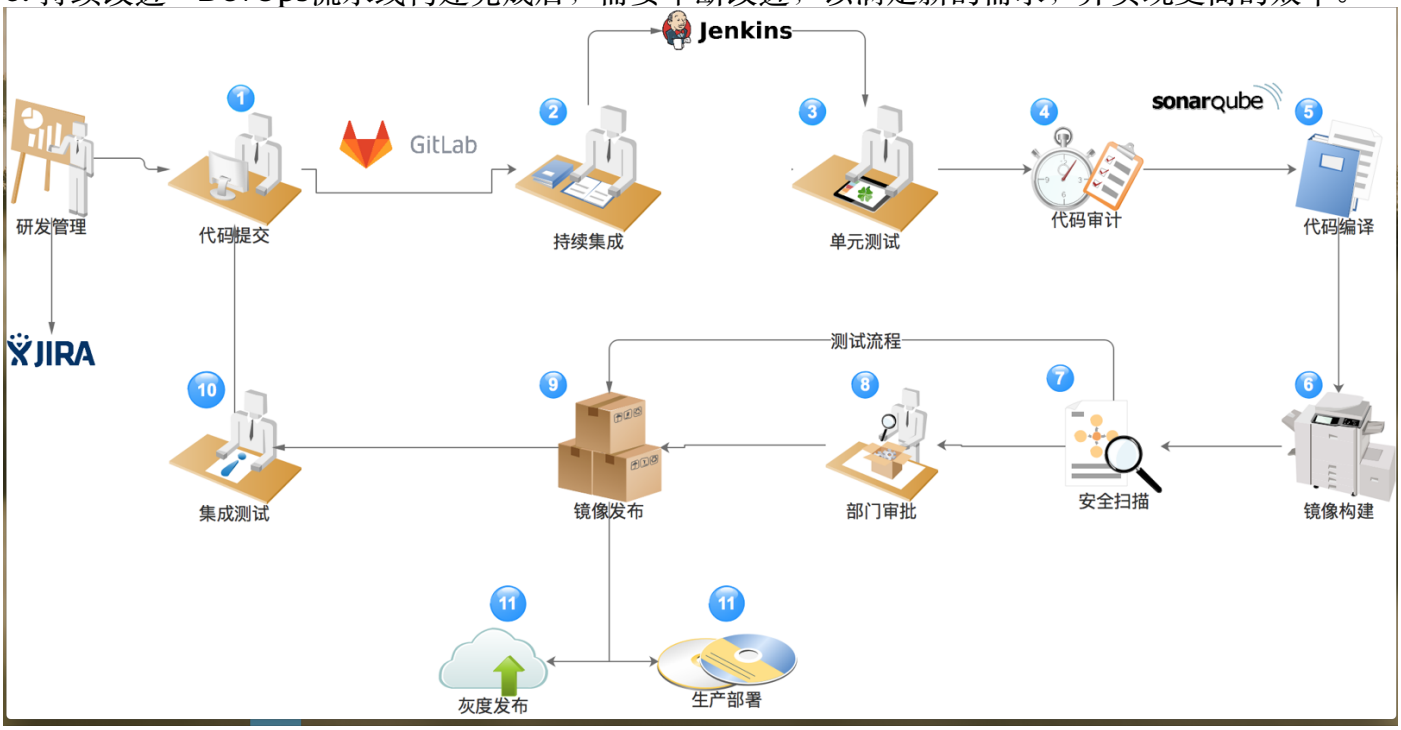
洪师傅，接着说，都是一点点积累过来的啊，希望可以帮助到你们，避免踩更多的坑。接着啊，我们还要学会分析，这就是最后一步：“善分析”，测试做到了各个环节，在做的过程中，我们能发现各种问题，通过收集到的这些数据，问题，我们要一点点的进行分析，善于分析，得出症结所在，然后对症下药，不断改善，持续精进。唯有这样子，我们才能不断的成长，为组织的发展带来坚实的信心和能力。测试急不来，也不能急，我们只有一点点做好，覆盖的到位，才能更好的为质量保驾护航。在追求测试的完美性这条路上，我们始终在前行，并且也将作为我们的行业追求，不断的发展，开拓。

而随着测试的深入，我们也可以追求更加高效和敏捷的方式，让测试取得更大的效果，比如，进行流水线的搭建，结合CI、CD，将开发、测试、集成和部署串联起来，让组织的运行效率达到更高的层次。不仅赋能组织，更要赋能业务。在这简答说一下流水线的搭建，这块涉及到DevOps。

DevOps是一种技术实践，旨在提高开发团队的协作效率，加快产品的交付速度。它涉及到各个领域，包括软件开发、测试、运维和云计算等。

要构建一个DevOps研发流水线，需要以下步骤：

1. 定义DevOps目标：首先，需要明确DevOps的目标，比如提高开发效率、缩短交付周期等，以便确定DevOps流水线的构建方向。
2. 构建DevOps环境：构建DevOps环境需要搭建软件开发、测试、运维和云计算环境，这些环境可以通过容器化、虚拟化、云计算等技术实现。
3. 集成工具：在构建DevOps流水线之前，需要选择合适的工具，比如持续集成工具、代码检查工具、部署工具等，来支持DevOps流水线的构建。
4. 构建DevOps流水线：根据DevOps的目标，以及上述环境和工具，构建DevOps流水线，以实现自动化的软件开发、测试和部署。
5. 运行测试：在流水线构建完成后，需要运行测试，以确保流水线的正确性。
6. 持续改进：DevOps流水线构建完成后，需要不断改进，以满足新的需求，并实现更高的效率。

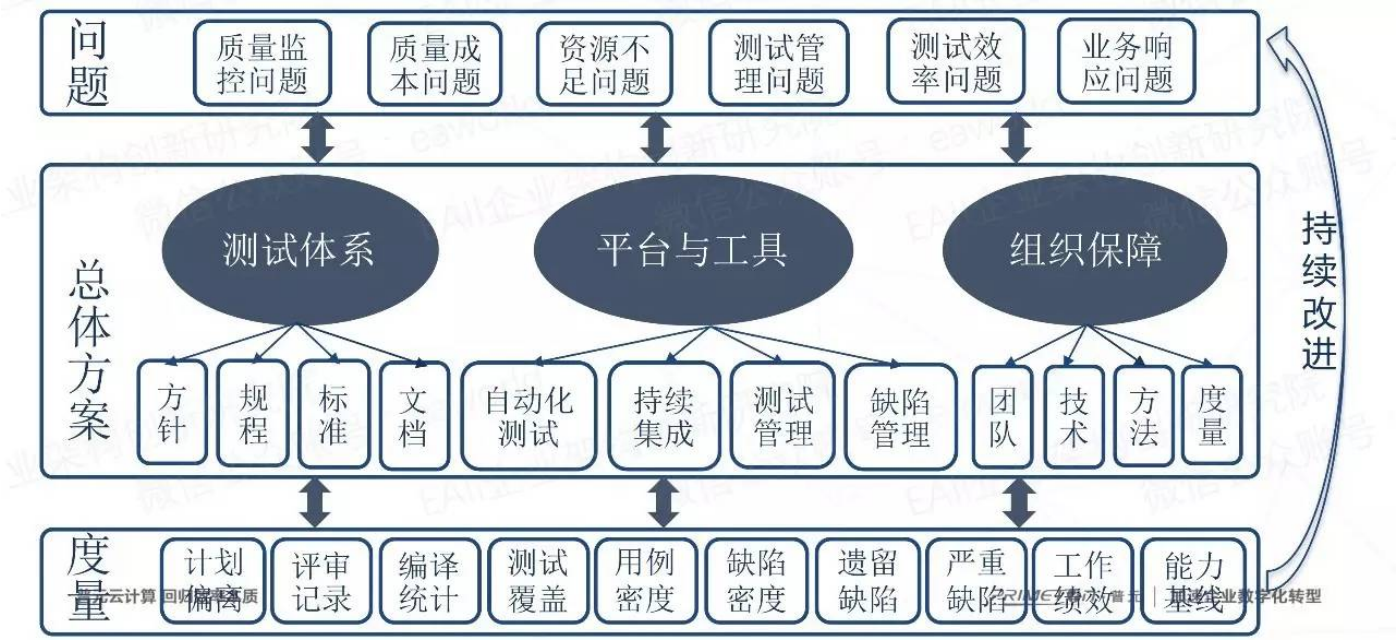


通过以上流水线的搭建，相信我们的研发效能和测试追求，可以达到更高的纬度。也为组织的变革管理提供了强有力的支撑。

所以在追求100%测试的这件事上，我们一直在路上，也将一直做下去。相信，最终我们将会受益良多。为用户提供体验更好的产品，不断提升生活幸福感，满足感。总的来说可以根据组织的实际情况，建立全面的持续测试管理体系，就我们的经验来讲，主要可以从测试体系、平台与工具和组织保障三个维度入手，有计划有步骤的开展实施，进行系统化建设。

1、首先，建立适合自身的测试体系：强调领导作用，制订质量方针；制定测试过程与规范；制定测试标准，准备过程文档模板；在研发团队内部进行宣导，由质量部门监督实施。 2、其次，选择适

合自身发展的测试平台与工具：引入持续集成与自动化测试平台，搭建测试管理系统，对测试需求、测试计划、测试环境、产品缺陷等管理； 3、再次，提供强有力的组织保障：提高质量是整个研发团队的一致目标，我们要从团队建设、技术能力、测试策略方法、监控度量技术等方面进行保障，并在每个项目的迭代种针对度量结果进行分析、总结，推动测试测试体系的不断改进的良性循环。



黎生听完之后，兴奋地喊了起来，这一套关于测试的方法手段，真的妙啊，从小的单元测试，到集成测试，在到新的测试手段，自动化测试，以及发布之后的遗留测试，在加上CI、CD方面的助力，让测试的整个过程都闹闹的抓在了我们的手中，而且通过这些方法，我们得到过程数据，在不断地分析，改善，持续精进。既有了不断前进的动力源，也有了持续改善的方法论，更有了组织变革的创新力。我们做不到100%的覆盖测试，但可以追求100%啊。谢谢洪师傅，今日听您一番话，让我醍醐灌顶，也看到了自己不足。希望回去之后，可以把今天您说的这些，逐步开展，一一尝试，结合实际情况，找到适合自己团队的平衡点，并持续改善。

洪七公说道：你说的很对，理解很到位，很通彻，是一个练武之才，不枉我今日与你沟通一番。你小子好好学，相信可以搞出一点名堂，有问题了在来找我。我还有事，就先行一步，说罢，便起身飞走，不见了踪影。

黎生回去之后，带领团队不断实践，持续改善，取得了很多的进步。然而，为了一己私利，杨康却命令黎生对欺压百姓的裘千仞道歉，但是黎生不畏强权，最终选择了自刎而死。虽然他是一个小人

物，但他坚持心中侠义的做法，值得侠义二字，坚持自己心中的侠义观，不向恶人屈服，即使只学会了一招，却依旧值得被人尊敬。所以，我们要根据自己的能力来承担起自己应该承担的责任，侠之大者，为国为民。

信息科技增长对公司的影响
移动信息化趋于国际化，对质量整体要求大大提升，重视质量，看重测试，对测试人员要求越来越高

公司业务发展对移动产品的影响
业务增速加快，移动信息化需要快速响应市场，要求产品更新和发布达到更快更稳定的程度

行业自动化测试发展的影响
软件巨头都有自己的自动化测试体系和平台（IBM、微软等）；移动行业处于快速发展期，商业软件->开源软件，自动化测试的重视程度越来越高

公司领导对测试团队有更高的期望
整体测试能力提升，提高测试效率；个人也希望提高对自动化测试认识，提高个人技能水平、满意度和归属感。

如同黎生一样，虽然只是一个小人物，但是依然对工作，生活充满了热情，有自己的大追求，大梦想。只有敢想，敢做，人生方能有收获。时间的长河里，困难重重，我们要做自己的主人，掌握主动权，及时出击，才能收获不一样的精彩人生。

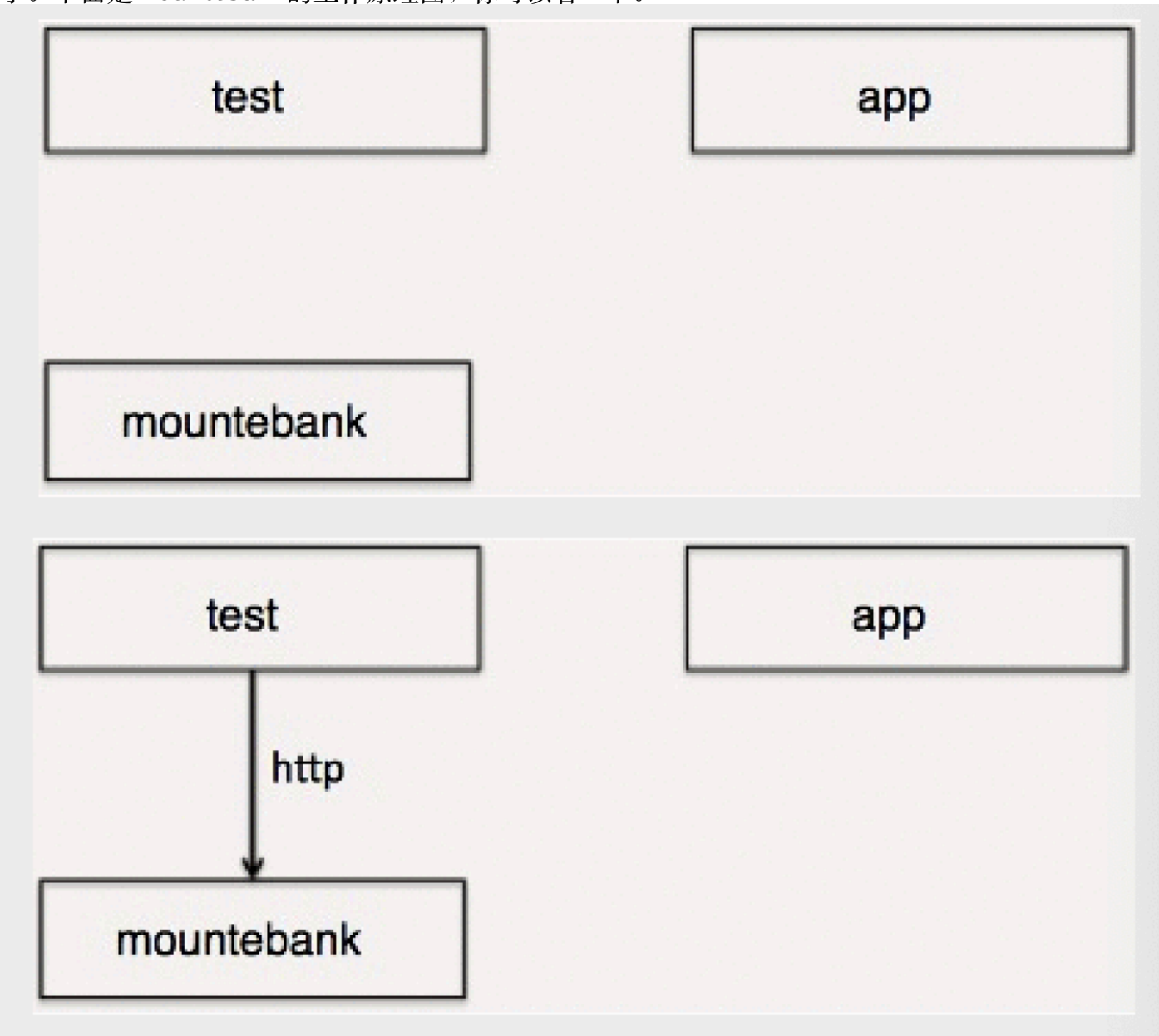
黄鹏飞
敏捷实践者，资深PMO

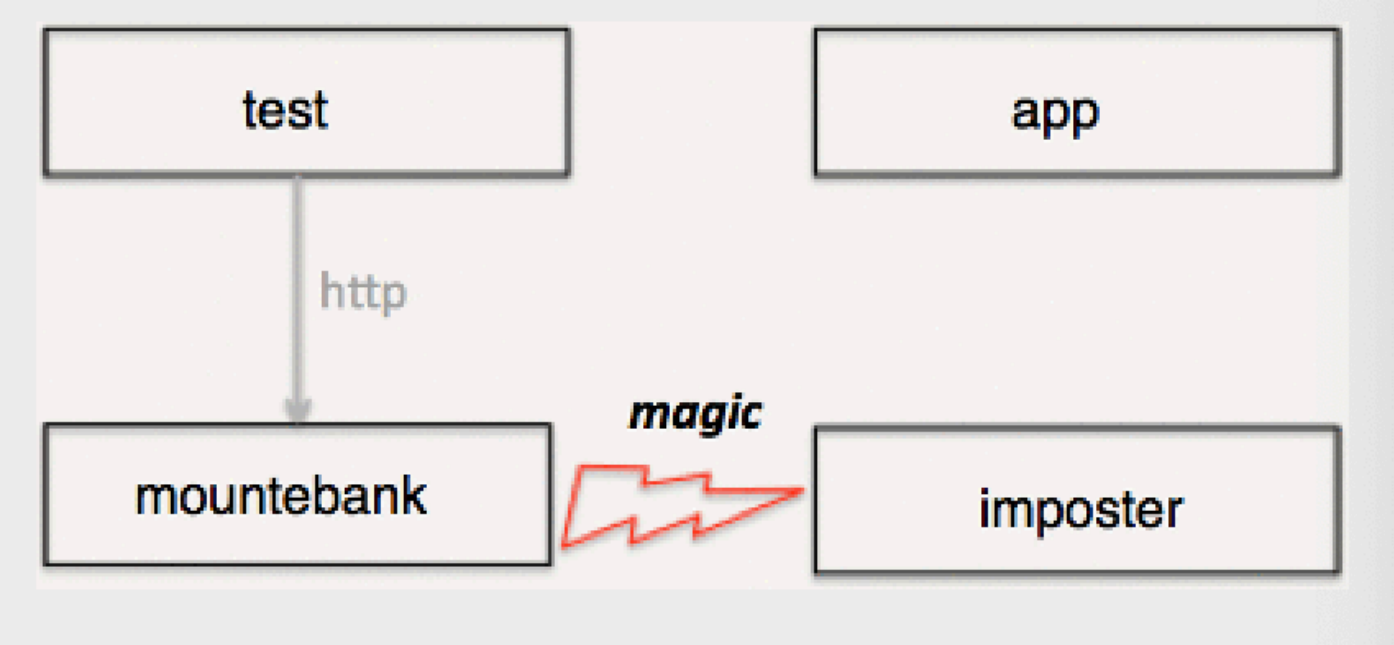
第6章 郭靖：降低服务间依赖，让自动化测试运筹帷幄

郭靖的《九阳真经》项目已经运行一段时日了。各个方面运行平稳。作为Scrum Master的郭靖看着团队每天高效的运转，心中开心不已。这一天团队中的资深开发鲁有脚在早会的时候提出一个问题，由于最近新加的一个功能是和全真教的北斗七星会议室做集成。最近网络原因，导致每次跑集成测试都比较耗时；另外账户有API调用限制，每天只能调用100次，但是随着自动化测试频率的上升，很容易达到这一限制的上限，就无法继续测试了。
郭靖听完一时没有头绪，于是安抚团队，承诺他会后去想办法。

会后郭靖独自来到了洞庭湖边，心想：团队最近几次迭代已经把敏捷开发的流程和和合作方式运用自如了。现在更核心的工程实践需求开始逐步体现出来。蓉儿曾经说过：敏捷开发最本质还是需要团队的工程实践更敏捷，否则就像地基不稳的空中楼阁。今天鲁有脚的问题，应该是团队开发中经常遇到的一个场景--依赖。服务依赖、组件依赖、产品依赖、团队依赖等。开发效率会随着依赖的增多、持续而逐渐降低。因为团队的很多工作要等待依赖项的完成。而这个是一个高度不确定因素。
想到此处，郭靖抬头眺望远处水天一色之处，大喊一声：“敏捷江湖，能否给我一点提示，有何妙招可以降低依赖？”话音刚落，郭靖突然感觉后背被一个石子击中。回头一看，原来是蓉儿。 只见黄蓉乐呵呵从远处走过来道：“靖哥哥，大白天对着湖面喊什么呀？傻乎乎的。”
郭靖眼前一亮，忙问到：“蓉儿，你来的正是时候。我遇到了如何降低依赖的问题，帮我出出主意。”随后郭靖就把鲁有脚早会说的事情复述了一遍。黄蓉听后点点头。神秘的笑笑说：“靖哥哥，我前几天去桃花岛见我爹，看到他的团队在使用两个工具，应该能帮上你。”
郭靖一听抱起黄蓉转了一圈。你可真是我的好蓉儿，快说来听听。
两人找了一个湖边凉亭坐下。黄蓉娓娓道来。

首先我听说一个工具叫做mountebank。是做打桩服务（Stub）的。它是业内著名的ThoughtWorks的工程师开发的。提供了多种协议的默认支持，例如：http，https，tcp，smtp。还有社区支持的idap，grpc，graphql，websockets等协议，像鲁有脚提到的是最常用的REST API调用，属于默认协议。你只要配置一些参数，就能够将实际的API请求拦截在打桩服务中。无需真的发送请求到第三方服务。这样自然对服务进行了隔离，降低了依赖。无论是网络要求和请求次数的限制都不存在了。下面是mountebank的工作原理图，你可以看一下。





mountebank会host多个imposter，都是根据实际需求配置的。当然打桩服务的一个主要目的就是，对实际代码不具有侵害性。所以你需要在路由层面做一个实际API请求域名和imposter的映射，让请求可以发送到imposter。这样你就可以根据测试需要，设定返回内容，甚至制造异常。咳，我给你写一个最简单的http请求的响应配置：

```
curl -i -X POST -H 'Content-Type: application/json' http://localhost:2525/imposters --data '{
  "port": 4545,监听端口
  "protocol": "http", 监听协议
  "stubs": [{
    "responses": [
      { "is": { "statusCode": 400 }}
    ],
    "predicates": [{
      "and": [
        {
          "equals": {
            "path": "/test",
            "method": "POST",
            "headers": { "Content-Type": "application/json" }
          }
        },
        {
          "not": {
            "contains": { "body": "requiredField" },
            "caseSensitive": true
          }
        }
      ]
    }]
  }]
}'
```

不符合条件的所有请求返回值

返回200 statusCode的条件判断

当你注册了这个stub信息到mountebank之后。我们可以看看实际的效果。例如：
○ 发送这个请求 curl -i -X POST -H 'Content-Type: application/json' http://localhost:4545/test --data '{"optionalField": true}'
你将会得到如下的响应
HTTP/1.1 400 Bad Request
Connection: close
Date: Sat, 24 Dec 2022 02:48:16 GMT
Transfer-Encoding: chunked
○ 发送这个请求 curl -i -X POST http://localhost:4545/test --data '{"optionalField": true}'
你将会得到如下响应
HTTP/1.1 200 OK
Connection: close

Date: Sat, 24 Dec 2022 02:48:16 GMT
Transfer-Encoding: chunked
是不是很容易上手？当然mountebank还有一些高级功能，例如：模拟网络延迟；通过javascript添加简单逻辑设定动态返回值；代理转发，多服务器集群等等。
对于个人使用打桩服务可选择的也挺多，例如：Postman的mock server，Fiddler的代理AutoResponder功能。不过mountebank可以用容器搭建一个独立的服务集群，给团队提供一套可重用和更高效打桩服务。
郭靖听的心花怒放。蓉儿真的帮了我大忙了。这个工具真是我的及时雨呀。
“蓉儿，刚才你说还有一个工具，岳父大人的团队也在用，能一起介绍一下吗？”郭靖道。
“当然靖哥哥。另一个是Mock工具，用在单元测试中。可以用于隔离测试对象依赖的其他类或者接口。”黄蓉回答道。“Mock工具会涉及到具体编程语言，因为要针对单元测试case编写对应的Mock部分的逻辑代码。所以Mock框架会和具体语言相关。这也是和打桩服务的不同点之一，打桩服务属于服务层面的模拟，和具体开发语言无关。靖哥哥，我爹他们团队用的是Java，所以他们在用Mockito这个框架，你的团队是用的是.Net开发，那么可以使用Moq这个框架。当然市面上还有很多其他类似的框架。你可以再调研一下。这些框架都可以很好的集成单元测试框架，例如：JUnit，xUnit等，用来编写单元测试。”
接着，黄蓉拿出两张羊皮纸，神秘的笑着说：“靖哥哥，你看这里有一个代码例子，是描述如何应用Moq的例子，我前一阵从欧阳锋那里顺手拿到的，嘻嘻。”
“例如我想测试下面controller里面的Index方法。我需要先想办法mock掉真实的数据库操作，因为我只需要测试其业务逻辑。”

C# 复制

```
public class HomeController : Controller
{
    private readonly IBrainstormSessionRepository _sessionRepository;

    public HomeController(IBrainstormSessionRepository sessionRepository)
    {
        _sessionRepository = sessionRepository;
    }

    public async Task<IActionResult> Index()
    {
        var sessionList = await _sessionRepository.ListAsync();

        var model = sessionList.Select(session => new StormSessionViewModel()
        {
            Id = session.Id,
            DateCreated = session.DateCreated,
            Name = session.Name,
            IdeaCount = session.Ideas.Count
        });

        return View(model);
    }
```

mock掉这个方法

“实际的Mock方法就像下面这个样子哦。”

C# 复制

```
[Fact]
public async Task Index_ReturnsAViewResult_WithAListOfBrainstormSessions()
{
    // Arrange
    var mockRepo = new Mock<IBrainstormSessionRepository>();
    mockRepo.Setup(repo => repo.ListAsync())
        .ReturnsAsync(GetTestSessions());
    var controller = new HomeController(mockRepo.Object);

    // Act
    var result = await controller.Index();

    // Assert
    var viewResult = Assert.IsType<ViewResult>(result);
    var model = Assert.IsAssignableFrom<IEnumerable<StormSessionViewModel>>(
        viewResult.ViewData.Model);
    Assert.Equal(2, model.Count());
}
```

mock测试需要的方法

将mock对象传入测试类

可以控制mock方法的返回值，用于断言判断

说完黄蓉看着张着大嘴，一脸惊讶的郭靖，打了一个响指笑到："靖哥哥，我说的这些够不够用？"
郭靖忙合上嘴道："够，太够了。待我消化一下。太感谢蓉儿了。"
之后几天，郭靖拉着鲁有脚等团队中的几个资深开发，利用业余时间学习了这几个框架。基本确认能够解决团队遇到的依赖问题。在接下来的两个迭代中，团队分别在单元测试和集成测试中分别应用了Mock和Stub服务，来降低依赖关系。不仅让测试效率得到了提升。测试的丰富程度也提高了。以往一些特定场景的异常处理也能够轻松模拟。
团队在回顾会上又总结了一些心得：
1. 打桩服务不仅仅可以应用在集成测试方面，例如压力测试的时候，如果想了解应用程序自身的性能问题，通过打桩服务隔离掉第三方服务。可以更好的确定性能压力的问题范围。提高解决效率。
2. 通过增加Mock和Stub服务，团队可以设定更多的测试场景，例如：特定异常的处理，网络延迟等。单元测试和集成测试的覆盖面更广，更可控。
3. 团队还整理了众多在测试模拟领域的各个名词。例如：fack，mock，stub，test double众多名词概念傻傻分不清。这里给大家引入一个微软开发文档中的解释，简单易懂，希望对小伙伴们理解他们有所帮助。
Fake - Fake 是一个通用术语，可用于描述 stub 或 mock 对象。 它是 stub 还是 mock 取决于使用它的上下文。 也就是说，Fake 可以是 stub 或 mock。
Mock - Mock 对象是系统中的 fake 对象，用于确定单元测试是否通过。 Mock 起初为 Fake，直到对其断言。
Stub - Stub 是系统中现有依赖项（或协作者）的可控制替代项。 通过使用 Stub，可以在无需使用依赖项的情况下直接测试代码。 默认情况下，存根起初为 fake。
郭靖开心的写了一个纸条放入漂流瓶扔进敏捷江湖：
巧用工具降低服务间依赖，让自动化测试运筹帷幄。

王东喆
敏捷教练
长春敏捷社区组织者
中国DevOps社区志愿者

参考引用:

1、http://www.mbtest.org
2、https://learn.microsoft.com/zh-cn/dotnet/core/testing/unit-testing-best-practices
3、https://learn.microsoft.com/en-us/aspnet/core/mvc/controllers/testing
4、https://learn.microsoft.com/zh-cn/dotnet/core/testing/unit-testing-best-practices

第7章 段誉：小步迭代价值流动，持续集成大显神通

大哥，大哥……”

“乔大哥！”

乔峰循声望去，话音未落，只见段誉以闪电般的速度又非常轻盈缥缈的箭步迎了上来。

“三弟，何事这么匆忙，不会又是版本发布向我求助，使用降龙十八掌和太祖长拳强修流水线吧？”

“不用啦，不用啦，这次不用啦！”段誉虽然跑得极快，气息却平稳轻松。

“刚才这轻功如此了得，上次见你还慢手慢脚的，跑得快一点就气喘吁吁的，这才三个月不见，功力长进不小啊”，乔峰观察道。

“哪里哪里，其实这次来找大哥是道谢的，以前要不是大哥鼎力相助，每次大版本发布之前强力修复集成流水线，以深厚的功力将所有自动化用例跑通，否则将导致版本发布不了产品功能上不了线，可能早就被研发老大鸠摩智炒鱿鱼了。”段誉这次一改往日愁容，整个人状态都不一样了。

“走！三弟特意带了大理国公元前1082年的陈酿，今天和大哥喝个痛快，一醉方休！”

话语间说着说着，段誉和乔峰就来到了附近最好的一家客栈“松鹤楼”。

“小二，上几个拿手的好菜！”

“好嘞，二位客官，您慢用”，段誉和乔峰一边大口吃酒，一边回忆起了之前发版本人肉暴力集成的往事。

由于系统组件配套依赖，环境复杂，用例众多，导致集成流水线年久失修，但多归多、杂归杂，所有人都知道，若要发版本，必须要有基本的质量保障。因此，流水线平时维护不起来，但在发版本的前两周也都是要咬着牙蛮力强修的，这需要每次发版本之前都投入大量的重复人力。

数月前，段誉痛定思痛，茶饭不思，下定决心一定要想出来个法子来。百思不得其解，有一天想的极其郁闷，就走出屋子去野外踱步，正思考的入神之时，一不小心脚底一滑掉进了一个山洞。

大概是山洞太高，跌落下去便昏迷了过去，过了大半天，段誉发现自己躺在山洞里。拍了拍身上的灰尘，起身打量着这个山洞，山洞里面有个石凳，石凳上面有一帛卷。

帛卷尽处题着“持续集成”四字，其后绘的是无数节点和箭头，注明“自动化”、“集成”、“流动”等等字样，尽是价值流动中的术语。段誉前几日还正全心全意的钻研自动化，一见到这些名称，登时精神大振，便似遇到故交良友一般。只见价值节点密密麻麻，不知有几千百个，自一个价值点至另一个价值点均有绿线贯串，线上绘有箭头，料是一套繁复的价值流动法。最后写着一行字道：“小步迭代，价值流动，持续集成，大显神通。”

段誉看得入了迷，如获至宝，真是“皇天不负有心人”呐。按照帛卷上所说，每日早午晚操练三次，每次小步集成，进步飞快。

卷到卷轴末端，又见到了“持续集成”那四字，登时便想到武林前辈马大师（Martin Fowler）对持续集成的定义：持续集成是一种软件开发实践，即团队开发成员经常集成他们的工作，通常每个成员每天至少集成一次，也就意味着每天可能会发生多次集成。每次集成都通过自动化的构建（包括编译，发布，自动化测试）来验证，从而尽快地发现集成错误。

段誉回去后，马上就召开了研发团队小组会议，立马就将习得的“小步迭代，持续集成”理念分享传授于每个人员，并倡导尽快应用于实践，并称这将会大大提升每个人员的幸福感，不再用版本发布前加班加点，这些事情做在平时。会议的主要内容是：

每个研发工程师都会将自己的代码提交到代码服务器中，项目组中的持续集成服务器。

从代码服务器中拉取代码进行构建，并进行自动化测试，最后持续集成服务器会将结果通知项目组成员，如果测试失败，第一时间进行问题定位及代码修改，持续集成服务器可以手动启动或者自动化定时启动。

一般情况下，建议项目组在每个工作日结束之后定时启动构建，通过日编译版本把控研发质量。

同时，制定了持续集成“军规”，并和研发部门秘书MM说将其打印出来宣传，发到每位研发人员手中，不要太单调，可以多样化，比如印到鼠标垫上，作为月度之星奖励。

持续集成的基本原则

- 只维护一个源码仓库
- 自动化 build
- 让你的 build 自行测试
- 每人每天都要向 mainline 提交代码
- 每次提交都应在集成计算机上重新构建 mainline
- 保持快速 build
- 在类生产环境中进行测试
- 让每个人都能轻易获得最新的可执行文件
- 每个人都能看到进度
- 自动化部署
- 构建失败之后不要提交新代码
- 提交前在本地，或持续集成服务器，运行所有测试
- 提交测试通过后再继续工作
- 回家之前，构建必须处于成功状态
- 时刻准备着回滚到前一个版本
- 在回滚之前要规定一个修复时间
- 不要将失败的测试注释掉
- 为自己导致的问题负责
- 测试驱动的开发

持续集成需要遵守一定的基本原则，左侧部分可以说是一些初级原则，右侧部分是对项目组更全面的原则要求，是完成持续构建必不可少的实践：

▫ 构建失败之后不要提交新代码——持续集成的第一禁忌就是构建已经失败了，还向版本控制库中提交新代码。如果某人提交代码后构建失败了，他就是修复构建的最佳人选，应尽快找出失败的原因并修复构建。

▫ 提交前在本地，或持续集成服务器，运行所有测试——不断的增量提交代码虽然是件轻量级的事儿，但确是一件严肃的事儿，我们需要在本地进行完善的自动化测试，尽量确保服务器构建的成功，不至于影响其他人的及时提交。

▫ 提交测试通过后再继续工作——提交代码触发构建后，开发人员应该监视这个构建过程，直到构建成功，构建结束之前，不应该被开会或其他事儿干扰。

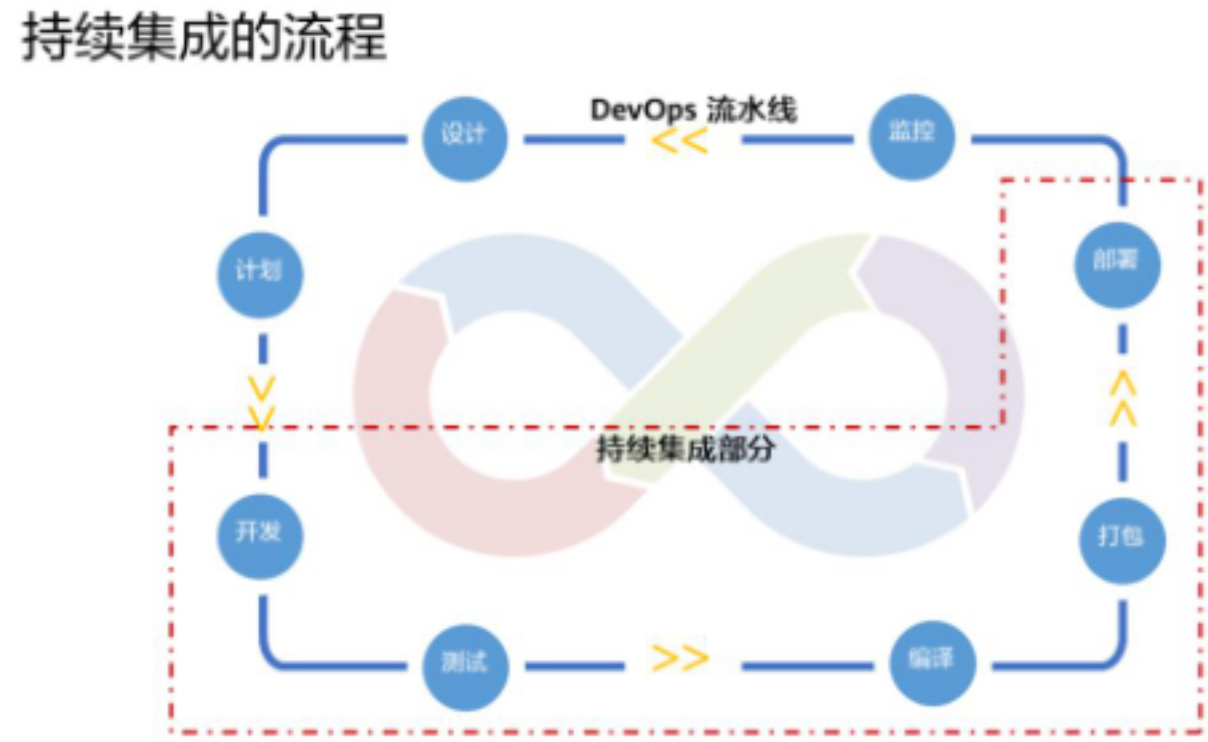
▫ 回家之前，构建必须处于成功状态——这是我们协同研发应该具备的基本个人素质，不让自己的工作成为团队的阻塞。

▫ 时刻准备着回滚到前一个版本——如果某次提交失败了，最重要的是尽快让一切再次正常运转，当无法快速修复问题，应该将它回滚到上一个版本。

▫ 在回滚之前要规定一个修复时间——对于是否应该等待继续问题定位，还是回滚版本，项目组之间应该有个约定，20分钟或是更久，我们需要有个时间界定。

持续集成的价值

在采用持续集成之前，软件项目主要面临几个问题：第一：个人环境开发构建，软件包差异较大，缺少一致的可部署软件，第二：缺陷暴露较晚，在研发后期发现的问题，一般需要投入更大的人力成本去修复，这也导致了其他的问题发生，软件产品质量较差，项目缺少可见性。
当团队采用持续集成之后，团队能够经常合并较小的变更，他们将减少合并的复杂性与努力，将减少变更到生产环境的前置时间，可以在任何时间地点从服务器轻松的获取可以部署的软件版本进行下一步动作，提高项目的可见性。



酒过三巡，段誉一五一十的回忆着，向乔大哥陈述着，个中的辛酸苦楚，只有经历过的人才能体会。传统的版本发布方式已不再可取，如今武林后辈后生可畏，新的工具层出不穷，每个团队都要与时俱进。

有了“持续集成”大法，段誉心中的一块石头终于落了地，乔峰作为大哥甚是高兴，两人喝了一碗又一碗，对饮数千杯，喝它个痛快。

好一个十六字诀，“小步迭代，价值流动，持续集成，大显神通”，妙哉妙哉！

刘志超

云原生、云服务架构设计&开发

中国DevOps社区核心组织者、精英译者

第8章 江南七怪：利用DevSecOps框架积极防御，构建安全可信的软件产品

黑风崖上白雪皑皑，寒风凛冽，大地一片萧瑟。

近日，江湖上突然传闻一种代号为XG的神秘病毒肆虐。谁也不知道这场危机由何处发起，但是已知其袭击了全国多个门派的软件产品，导致线上系统纷纷瘫痪，损失惨重。一时间，江湖谈其色变，人心惶恐，这注定不是个平静的冬天。

在黑风崖山脚的一处避风的山庄内，以柯镇恶为首的江南七怪正在积极商议如何构建安全屏障，保障系统在这恶劣的环境下能继续正常运行。

“各位兄弟姐妹，目前情况如何？”柯镇恶神情凝重地问道。

“大哥，想来那病毒已经对安全薄弱的门派下手了！这次江湖危机给大家敲响了警钟，软件产品研发不能为了追求速度而忽略了安全方面的关注。还好我们团队近年来积极探索优秀的安全研发解决方案，一直坚持敏捷持续交付和安全研发的理念，自从实施了DevSecOps框架，安全基本内功已小有建树。那病毒近日来对我们的系统发动过几波袭击，都被我们提前识别并及时清除了，我们对抵御日常的攻击还是有信心的。”老二朱聪看着柯老大说。

“是的！”老三韩宝驹金龙鞭一抖，底气十足地说道，“通过安全培训和文化建设，现在团队的安全理念有大幅提升，以往我们只能依靠个别的安全护卫找到问题再来想办法解决，那样响应速度太慢了，效率也不高。现在我们的团队已经把安全已经贯穿到了软件生命周期的每个环节，形成安全研发人人有责的研发氛围。不仅有安全团队，还有IT研发、运营、甚至业务都积极参与到安全保障行动。现在的IT团队应对安全需求设计、分析和解决问题等能力提高了，对安全需求响应速度大大提升。”

老四南希仁附和道：“没错！我们团队落地DevSecOps框架重要的关注点之一就是安全左移。业务、开发与安全团队在研发前期紧密合作，一起参与安全需求评审，主动识别安全合规风险，制定了整

合的安全方案。通过分析进行更智能的安全防护，一般的安全问题早发现早解决，避免后期返工，提升了交付效率。”

“在开发阶段，安全编码和利用工具自动化扫描漏洞也是我们团队DevSecOps的重点实践。我们开发团队有良好的开发编码规范参考。开发在编码完都要求完成代码Review和必要的自动化的源码漏洞安全检查才能集成代码。目前CI/CD流水线已经集成必要的安全扫描工具，通过自动化工具及时发掘问题，提高检测效率的同时也避免人为因素导致遗漏检测。我们摸索出一套标准化漏洞管理流程，将扫描出来的漏洞进行严重级和优先级标识，区分内部威胁和外部威胁，以及做好分类管理，对危重级别高的漏洞进行拦截告警。这些方法有助于团队快速处理关键问题。”老五张阿生说。

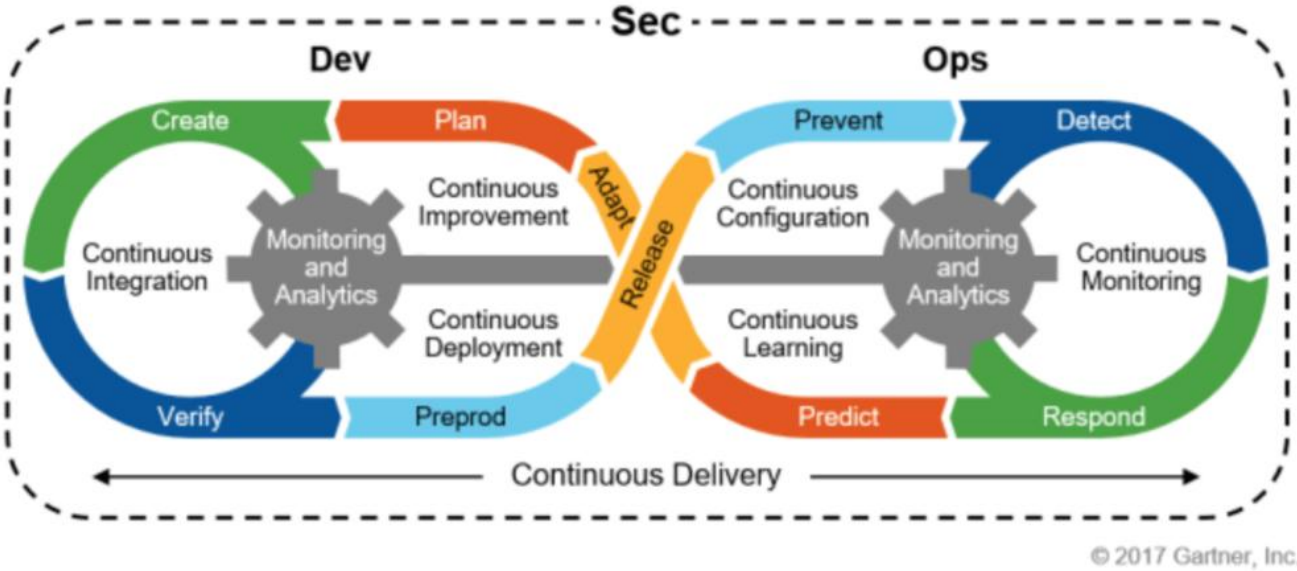
老六全金发点点头补充说：“我们还完善了开源和第三方组件的安全管理规范，利用工具扫描现有的开源和第三方组件漏洞对产品团队进行告警，对计划引入的开源和第三方组件进行合规审批，持续更新已知的第三方软件威胁和漏洞列表。”

“大哥，运营方面我们也做了一些安全方面的改进措施。实施了有力的安全访问机制，可追溯的变更管理流程，完善了安全问题的响应机制等；我们还会定期总结分析典型的安全问题形成案例库进行分享学习，提升整体安全意识和知识，避免类似问题重复发生。”性子活泼的七妹韩小莹接着说。

柯镇恶听到大家的讨论，心里放心下来，拄着拐杖郑重地说：“大家都做得很好! 打铁还得自生硬，大家切记不可大意。我们兄妹七人这些年坚持正义之路，通过运用DevSecOps框架进行产品研发方能在武林上有一席之地，靠的就是扎实的安全内功，安全可信口碑！事实证明，我们的选择是正确的。”

情绪激动的柯镇恶继续说：“在DevSecOps实践方面，我们还要不断探索，总结优秀的实践经验推广到其他的团队，让更多的团队受益。”他拱了拱手说:“谢谢大家了！”

此后，DevSecOps理念犹如一只定心丸在江湖广为传播。各大门派利用DevSecOps框架的优秀实践经验更科学有效地构建安全可信的软件，为终端用户利益保驾护航。



刘陈真
敏捷教练

第9章 黄老邪：武林债易躲，技术债难还

纵观武林，论技术，圈中不知道谁更高，或者不同语言和框架都最终殊途同归，但我知只有那数不清看不明的技术债，会在深夜中让你突然坐起，经久不眠。

黄蓉：靖哥哥看你这么愁眉苦脸的，是不是有什么烦心事啊？

郭靖：哎，虽然我拿到了《九阴真经》，《武穆遗书》，又坐了多啦大侠的时空门在多元宇宙中拿到了张无忌的《九阳神功》秘籍和无崖子的《北冥神功》。但还是无法偿还这一身的债。

黄蓉：哦？难道是你和华筝公主的情债？无崖子就是那位颠来倒去，飘来飘去的老爷爷吗？你说是不是他介绍你们俩认识的？哼~

郭靖：蓉妹妹，看来你还是不够懂我。作为武林编程盟主，我是想让诸多的编程经典无害化，让大家安心练武，不再出现像欧阳锋，以及梅超风那样走火入魔，天天996-007，做出败坏武林风气的事情。

黄蓉：哦？那人家错怪你了啦，说来听听看。

郭靖：通过我在多元宇宙的走访，我发现之所以出现996-007，是因为大家在编程的过程中只注重行为上的不断操练，缺忽略了意念的修为，从而积累了大量的技术债，最终以债换债在996-007中走火入魔。

黄蓉：技术债？是那些练武的技术宅借了P2P还不上吗？

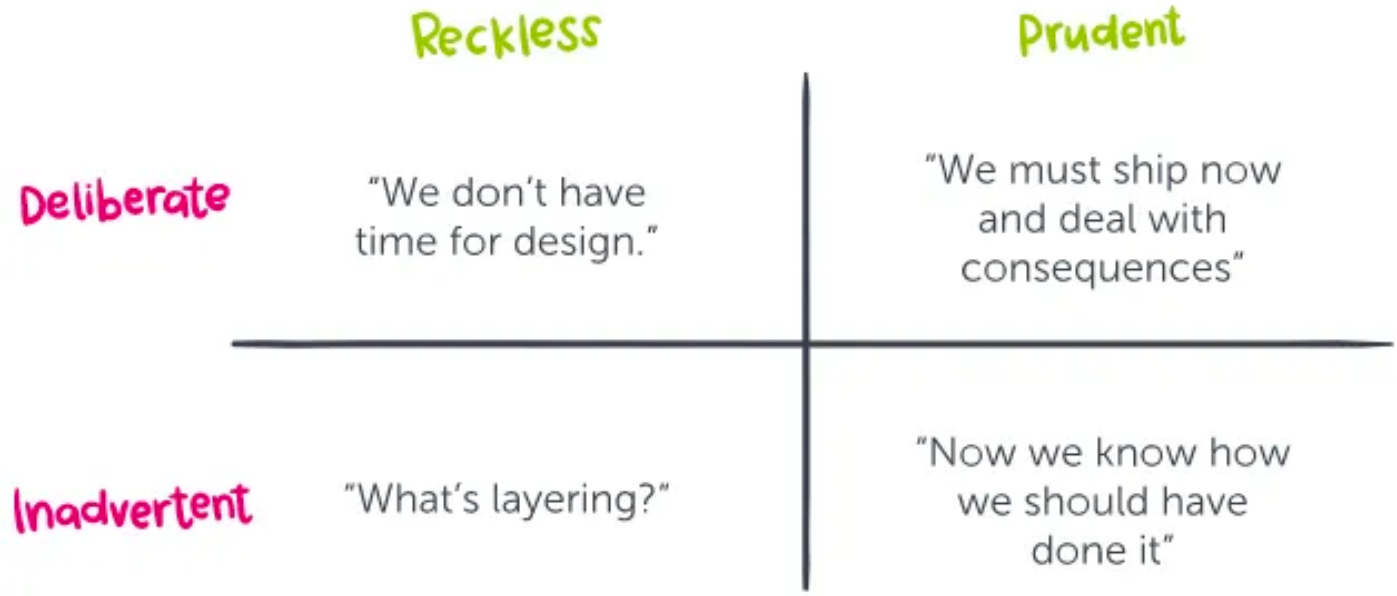
郭靖：我。。。!!! 技术债描述了在习武编程过程中有意或无意只注重效果（客户价值和/或项目限制，如交付期限），而忽略了意念的提升和修炼（技术实施和设计）”

黄蓉：好高深哦，你说的这些我也有诶，能具体点吗？

郭靖：

技术债1：遗留债。这点上，就不得不提老顽童了，他写的代码过于老旧，JDK6用了20年，所以遇见你父亲那种标新立异的leader就完全无法招架，写的代码完全无法运行。

技术债2：规划债。要说到规划，我看就属蓉儿你了，你冰雪聪明，博学多才，身怀绝技，但疏于规划，做项目经理只看眼前的利益，不做DevOps，到了项目后期才发现花在代码集成、代码分钟维护的时间几何级上升。
技术债3：重构债，要说一手好牌打的稀巴烂的当属过儿了，他学了你爹的的玉萧剑法、弹指神通；老毒物的蛤蟆功、移穴换位；师傅的打狗棒法、以及中神通的全真剑法，但他生性浮躁，对于自己写的代码疏于重构，博而不精，一生坎坷。
技术债4：人债，要说技术债中最具杀伤性的要属人债了，就像萧峰大侠用人不淑，惨糟奸人陷害，从一个武林帮派的CEO沦为一个外包人员。
黄蓉：靖哥哥你最近的代码武功长进好快啊！
郭靖：哪里哪里，这都是跟你爹爹学的。
黄老邪：哦？有人在叫我吗？啊哈哈哈。
黄蓉，郭靖：爹爹。
黄老邪：我刚才一直在听靖儿的分析，为父也觉得入木三分。我当年和老马（Martin Fowler）华山论剑时，听到他对于技术债的分析，深以为是。他将技术债以习武者的心态而分，以是否是主观意愿故意欠债为纵坐标，以是否认真规划为横坐标。蓉儿你也要像靖儿学习。



黄蓉：爹爹，蓉儿比不过，这不是给你找了个好女婿继承你的才学嘛。
黄老邪：武林债易躲，技术债难还。你呀~总之你们要用心些。注重武学编程经典规范，日三省乎己时时重构代码，有问题及时解决，这样才能有效避免技术债的积累，自身热爱，自我沉迷，这样才能有效避免走向996-007的魔道。
黄蓉：记住啦爹。

胡帅
软件工程专家
中国DevOps社区精英译者，优秀志愿者

关于社区

「为推动DevOps在中国的蓬勃发展，我们一群有着相同价值观与理念的人，携手发起了“中国DevOps社区”。」

社区使命：传播DevOps文化，落地DevOps实践

社区愿景：成为中国DevOps运动的领航人与催化者

中国DevOps社区成立于2018年，至今，我们已累计在22个城市举办109场DevOps Meetup、5场DevOps行业年度峰会，传播1000+篇DevOps实践案例，影响10万+软件研发从业者。

中国DevOps社区的治理架构由理事会、城市核心组织者、志愿者和社区成员4部分组成，共同治理社区秩序，发起社区活动，服务社区受众。由理事会支持的专职运营团队负责全国社区的运营，城市Meetup由全国核心志愿者共同组织，社区内容由志愿者群体自愿、自主、自发地分享与贡献。

所有社区成员共同坚守开放、专业、使命感的核心价值观，始终保持中立，保持开放宗旨，包容百家思想，建立伙伴关系，创造社区价值。

社区宣言

作为有理想的中国*DevOps*社区志愿者，我们一直身体力行，传播*DevOps*文化，落地*DevOps*实践，并帮助他人学习与实践*DevOps*。与此同时，我们建立了如下价值观：

不仅要让社区中立，更要 保持开放宗旨
不仅要去响应需求，更要 创造社区价值
不仅要有专业技能，更要 包容百家思想
不仅要与多方合作，更要 建立伙伴关系

也就是说，左项固然值得追求，右项更加不可或缺。

中国DevOps社区，与志愿者一起，为全国实践者提供一个可本地就近参与、活跃的线下/线上交流社群。

关注中国DevOps社区，你将与全国DevOps实践者共同成长；成为社区志愿者，你将为推动中国DevOps实践与发展贡献一份力量。

社区官网：https://DevOpsChina.org

扫码关注公众号

扫码报名志愿者

扫码关注B站号

扫码关注视频号

欢迎投稿

中国DevOps社区电子期刊《敏捷漂流记》欢迎投稿~

第一期：《敏捷站会的九大坑》 完结
第二期：《迭代回顾九大坑》完结
第三期：《需求管理九大坑》完结
第四期：《迭代计划九大坑》完结
第五期：《工程实践九大坑》完结
接下来的第二季启动~

任何建议 & 投稿，欢迎联系社区小助手

社区公众号：DevOpsMeetup
社区官网：https://DevOpsChina.org

守望者

敏捷漂流记

第四期 「工程实践九大坑」

出品：中国DevOps社区

作者：陈文峰、王英伟、李岩、方正、黄鹏飞、王东喆、刘志超、刘陈真、胡帅

排版编辑：成芳

任何建议 & 投稿，欢迎联系社区小助手~

社区公众号：DevOpsMeetup
社区官网：https://DevOpsChina.org

谨以此书献给中国*DevOps*社区的志愿者、粉丝、赞助商及生态伙伴们。

感谢大家伴随我们一路前行，一起建设社区，一起为*DevOps*在中国的推广所做出的卓越贡献。