工程能力

白皮书

工程标准V2.0

Baidu 百度

前言

自 1968 年软件工程被首次提出，到今年刚好是 50 周年。在这半个世纪的时间里，质量和效率是软件工程领域永恒的主题。人类自第一次工业革命以来， 解决质量和效率问题的不二法门就是自动化、自动化、再自动化。智能化更是自动化的一种高级形态。然而在当今企业中，这样的一个根本方法却被人们无意或有意的忽视。说无意忽视，是因为在企业中负责工程能力建设的人可能缺乏工程经验，心中眼中只有流程。仿佛流程顺了，即便大都靠人工执行，也心安理得。说是有意，则是因为软件行业中原有的分工等原因，自动化的导入带来分工模糊、岗位重要性变化等诸多不确定性，缺乏引入的意愿。本人无意贬低流程，而是更推崇自动化和智能化的流程，更无意非议职责分工，而是更推崇自动化和智能化解放人，使每个人都能更多从事创造性的工作。

人们对于人与世界、人与人的关系的认知，在过去的半个世纪里并无显著突破，这些认知又进一步决定了流程优化的上限，仿佛一道透明天花板，人人可感，却无从突破。另一方面，ABC 技术 (A-AI 人工智能、B-Big Data 大数据、C-Cloud 云计算) 突飞猛进、日新月异。如果工程能力的建设无法站在 ABC 技术进步的巨人肩膀上做创新，那是软件工程领域在这个时代的悲哀。

互联网公司关注快速迭代，天生具有追求卓越工程能力的基因，百度作为技术立身的企业，创立早期便启动了研发流程自动化的工程能力建设，需求流转、文档阅读、代码搜索、程序分析、软件测试、制品部署、服务监控、用户反馈等覆盖研发全流程的工具平台积累，涉及到 Server、App、SDK、AI 算法模型、自动驾驶等异构产品业务形态。40 余个工程实践环节，积累了 40 余个可插件化接入的平台，在公司内被广泛使用。在这些实践和平台运行的同时，一个新的金矿正在悄然生成，那就是软件工程大数据。如今我们依托百度领先的 ABC 技术和工程能力建设经验，将公司积累的 500 多亿条软件工程数据进行处理，其中锤炼出的技术用于代码推荐、缺陷检测、智能测试、智慧运维等领域．今年百度承接了国家重点专项《基于编程现场大数据的软件智能开发方法和环境》，与国内软件工程领域实力顶尖的 15 所高校一同建设中国未来的智能软件开发平台。

作为百度内部以『工程卓越、毕生所求』为使命的一群人，我们一边支持公司业务打胜仗，一边努力探索工程能力建设的新方向，心中充满着欣喜。并非常渴望能将我们的理解和主张向行业分享，共同进步。然而由于工程能力体系庞大、平台众多，一时无法穷尽，所以我们将这个体系进行分解，首先分享价值高、可迁移的《工程标准》。后面陆续会向大家汇报我们在 AI 赋能软件开发、AI 开发软件工程等方面的进展。更期待您的反馈和交流。

李涛

百度工程效率总监

百度平台化委员会秘书长

2018 年 10 月 10 日

编委会成员

目录

百度软件工程标准概述

工程标准制定的背景

百度是一家技术公司，软件研发工程能力的水平直接影响公司的持久创新力和公司在市场上的作为。只有不懈追求卓越的工程能力，才能够带来长期的核心竞争力，才能为每个用户、每个企业客户以及整个社会创造价值。长期以来，百度在大量的软件开发过程中尝试了多种方法和工具相结合的实践，并总结了许多优秀的工程实践。经过长期团队观察和大量研发数据分析，我们证明这些优秀实践可以有效帮助提高团队的软件开发效率和产品质量，应该把这些优秀实践的采用和实施推广到更大范围，因此制定百度软件工程标准。

工程标准的目标和意义

百度软件工程标准制定的目标是帮助研发团队持续提升工程能力。工程标准可以快速指导团队采用优秀的软件工程实践和研发工具，使其在研发效率或产品质量上获得提升。同时有了标准和规范，也能够更有效地衡量团队工程能力的水平，让各个团队能够更好地了解自身的工程能力现状，进而设定工程能力提升目标，不懈追求工程卓越。

本白皮书希望分享百度在软件工程标准、实践、度量和改进方面的经验，呼吁业界共同加强工程能力建设、研发工具投入、工程标准更新与工程素养提升，共同推进软件工程的发展。

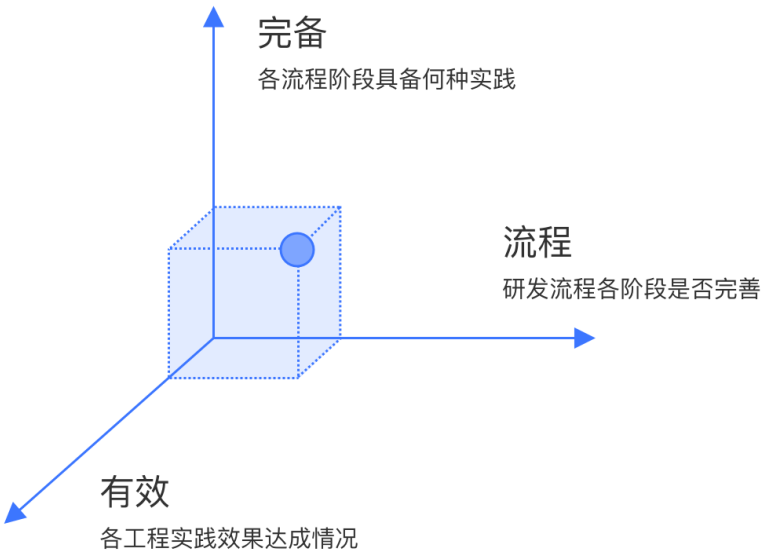
工程标准的修订规则

百度软件工程标准是由百度 DevOps TOC（Technical Oversight Committee）制定并发布的，并且随着公司工程技术的发展不断更新。首先，DevOps TOC 的委员广泛收集各个研发团队的优秀工程实践，以及工程标准实施的反馈，制定工程标准的初版或修改意见草案。然后 TOC 委员将草案提交至 DevOps TOC 进行充分讨论，如果 TOC 会议通过，就进行标准修改的公示，同步修改研发工具中的对应规则。同时收集各团队反馈和实际研发数据进行分析，来验证标准实施的效果，并继续准备下一轮的规则更新。

本白皮书是 2018 年《百度工程能力白皮书 - 工程标准 V1.0》的升级版本。V1.0 版之前，百度软件工程标准已经在百度内部经历了十余次修订。本次 V2.0 版本是 DevOps TOC 在 V1.0 版本的基础上，按照上述修订规则经过数次修订后发布的新版本。基本包含了百度所有产品团队的优秀工程实践，具备了可靠的权威性，起到了有效的指导作用。

工程标准的制定思路

百度软件工程标准是从公司大量的软件工程实践中挑选总结而成。主要参考依据为流程是否顺畅、实践是否完备、实践是否有效三个方面，再加上构建系统是否可靠稳定，形成四位一体的工程能力标准，已达到流程全面、实践完备、效果驱动、结构合理等特点。



百度软件工程标准新版介绍

工程实践规范标准升级

从 2018 年 10 月工程标准 V1.0 发布以来，截止到 2019 年 8 月，百度 DevOps TOC 在工程标准的实践集合中删除了 5 个实践，其中包括开发阶段的本地检查的三个相关实践，以及“重复文件检查”、“容量测试”两个实践。另新增了 5 个实践，分别为开发阶段的“第三方代码依赖”，“本地代码检查”；APP&SDK 测试阶段的“隐私合规测试”，“稳定性测试”；以及 Server 上线验证阶段的“容量评估”。相较于白皮书 V1.0 版，新版本的工程标准更新了 14 个工程实践的标准，使标准进一步细化并且更方便落地实施。

流水线及自动化规则调整

V1.0 版本的“流水线自动化”标准中从流水线“执行效率”、“失败恢复时间”和“异常构建率”三个指标项来描述流水线自动化的能力。但实践中发现团队在代码准入、测试、上线等活动中，会设置不同类型的流水线支持上述活动，由于这些开发活动的频率、构建时长、质量要求各不相同，用统一的三个指标来衡量不同类型的流水线，会导致无法有效衡量流水线自动化能力的情况发生，也无法有效指导流水线自动化能力的提升。

为此，在 V2.0 版本中，把流水线自动化的规则，从计算流水线整体健壮性分解为分别考察每个阶段中任务的自动化程度。

加入 DevOps 效果指标

采用工程实践的直观变化是工程实践完备程度和有效程度的提高，最直接的目的则是研发效率（速度）和研发质量的不断提升。在 V2.0 版本中，加入了对效率（速度）和质量的指标定义和度量方式，以便每个团队在不断提升工程能力的同时，也能够直观看到他们研发速度和质量的同步提升。

百度软件工程实践集

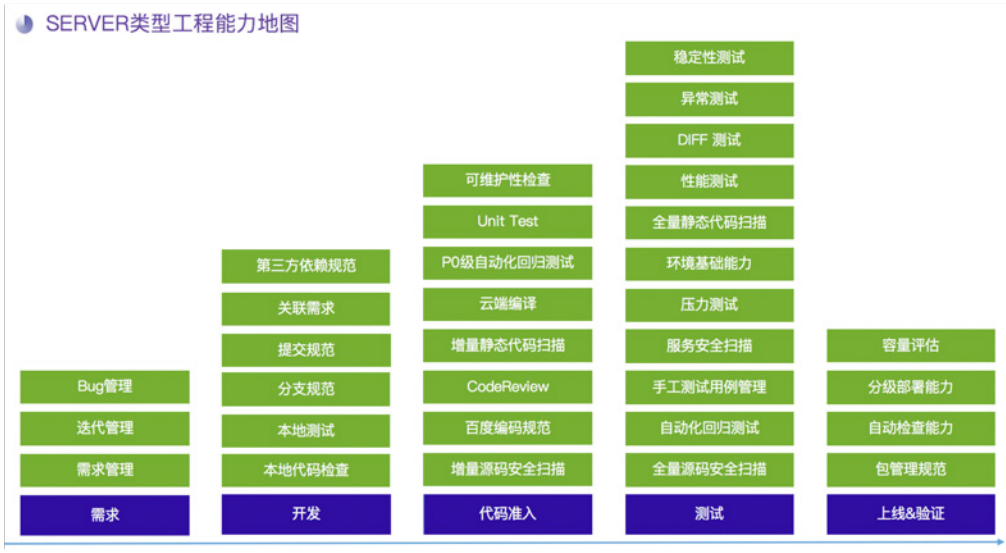
百度的软件工程类型

百度公司的软件产品形态有多种，例如 App，Browser，PC Client，SDK 等。不同的产品类型其研发过程及优秀实践也不尽相同。目前我们把研发的工程类型大致分为 4 类：Server、App、SDK 和 AD（Autonomous Driving），本白皮书介绍 Server、App 和 SDK 三种工程类型的工程标准。Autonomous Driving 工程类型的工程标准将在白皮书后续版本中增加。

Server 工程类型的工程实践集

开发一个 Server 工程，需要经过需求、开发、代码准入、测试、上线 & 验证等阶段，每个阶段有若干优秀工程实践。

下图是百度 Server 工程类型的优秀实践集合——Server 类型工程能力地图。通过工程能力地图，可以指导从开发到上线的流程，建立标准化研发工具链，统一度量团队软件工程的能力。

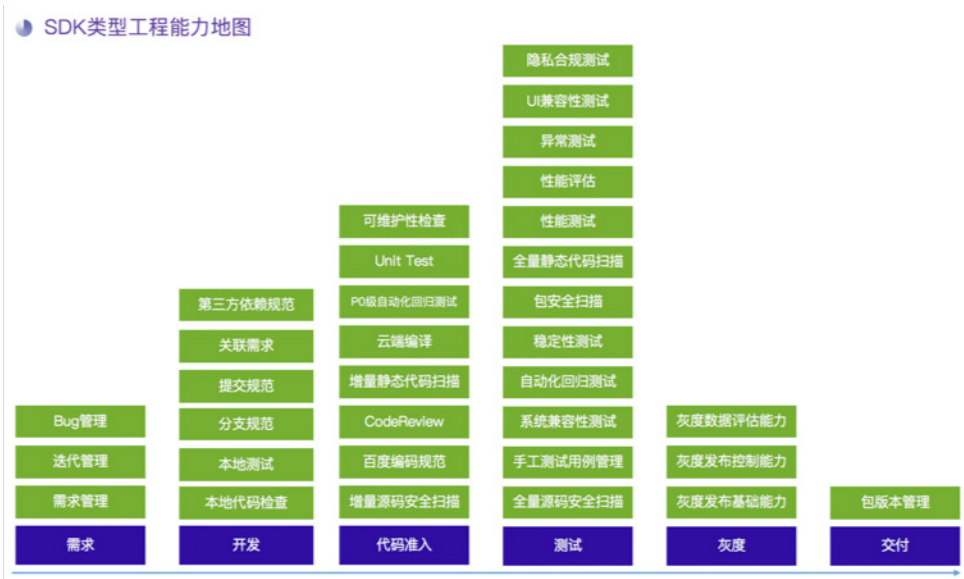


App 工程类型的工程实践集

开发一个 App 工程 ,需要经过需求、开发、代码准入、测试、灰度、发版等阶段。下图是 App 类型工程能力地图。

SDK 工程类型的工程实践集

开发一个 SDK 工程 ,需要经过需求、开发、代码准入、测试、灰度、交付等阶段。下图是 SDK 类型工程能力地图。



部分实践名词解释

【容量评估】

通常是通过对线上系统不断地增加压力，直到系统的某项或多项性能指标达到极限，以此来观察系统的整体容量。容量测试的主要目的是评估在真实业务流量情况下系统整体吞吐量的变化及其规律，以便指导运维和扩容等工作。

【异常测试】

通过模拟系统运行过程中出现的各种异常场景，如网络、硬盘、输入等，测试系统对异常的容错性，提升系统的鲁棒性。

【DIFF 测试】

通过向两个及以上的相同服务（可能不同版本）发送批量请求，分析返回结果，以验证系统可能出现的随机或返回不稳定等问题，或评估新服务策略升级效果等。

【性能测试】

这里特指狭义的性能测试，是指以性能预期目标为前提，通过对系统发送正常流量的请求，验证系统在资源可接 受范围内，是否能达到性能预期，以便发现性能问题。

【全量静态代码扫描】

通过引用丰富的工具对代码全库进行源码扫描，发现代码可能存在的风险或漏洞。

【环境基础能力】

需要不断提升被测系统与其线上系统的模拟真实性，含流量、拓扑、机器等。

【压力测试】

指在超过安全负载的情况下，在线下对系统或模块不断施加压力，通过确定瓶颈点，来获得能提供的最大服务级别的测试。压力测试的目标常见为模块。

【服务安全扫描】

对 web 或 api 类服务进行请求或者服务扫描，以便发现服务中出现的安全漏洞。

【手工测试用例管理】

合理的用例管理，可以让同一批人共享用例提升效率。

【稳定性测试】

指在特定硬件、软件、网络环境条件下，通过向系统加载一定压力并运行较长时间，以此检测系统是否稳定。

通常在测试过程中会执行一些常见的影响系统稳定性的场景。

【自动化回归测试】

用于对被测系统进行功能逻辑的回归验证，以便提前发现新策略的问题或新代码对历史功能带来的影响。

【全量源码安全扫描】

通过对代码全库进行扫描，以便发现源码中可能存在的安全漏洞，如显示密码等。

【Unit Test】

软件中的最小可测试单元进行检查和验证，不用搭建被测系统即可验证代码的逻辑或功能。

【P0 级自动化回归测试】

自动化回归测试的定义如之前所说，P0 级是指被测系统的核心功能点。

【CodeReview】

通过代码走查，发现代码的规范、风险等问题。

【百度编码规范】

代码符合百度 CMC 制定的代码规范。

【增量源码安全扫描】

对当次提交的代码进行源码安全扫描。

【增量静态代码扫描】

对当次提交的代码进行静态代码扫描。

【包管理规范】

发布上线包需要规范化、安全存储，也需要符合公司对包描述的管理规范，以便做到可追溯等。

【分级部署能力】

具备按照流量切分、环境隔离等逐步上线控制能力，供自动检查或快速回滚等场景使用。

【自动检查能力】

发布上线过程中 ,具备自动化功能检查、线上系统运行情况监控等能力 ,在系统最后上线过程中有效发现问题。

【本地测试】

开发者通过 API、脚本等发起功能、单测等相关验证能力，对代码在正式入库前进行验证。

【UI 兼容性测试】

通过各种类型的遍历方法或定向 case 集合，对执行过程中的截图进行白屏、乱码、屏幕兼容等校验，检查 UI 可能存在的问题。

【性能评估（APP）】

通过各类核心 case，统计 app 在 case 执行过程中的性能表现，以评估升级过程中带来的性能问题。

【性能测试（APP）】

通过检测代码，发现内存泄露、句柄泄露等风险。

【包安全扫描】

通过对 APP 包进行扫描，扫描出包中可能存在的安全性风险。

【稳定性及遍历测试】

通过对 APP 页面各类元素进行模拟点击，发现该过程存在的崩溃等问题。

【系统兼容性测试】

让 APP 在大批量不同手机、不同系统上进行安装、卸载、执行，发现 APP 的潜在问题。

【隐私合规测试】

为确保符合《个人信息安全规范》等法律法规和隐私政策的要求，同时符合公司《安全红线》和《隐私红线》的要求，对产品功能中涉及到个人信息数据采集、存储、访问、处理、共享、删除的敏感权限、使用场景、隐私数据、隐私政策等进行合规测试。

百度软件工程实践标准

说明：下面的三级标准里面，数字相关的要求是根据百度历史经验估出来的中位值，会不断持续更新，很多优秀团队的工程实践往往会优于这些值。

需求（Server/App/SDK）

需求管理

- Average　在 iCafe 中统一管理需求，记录需求内容和相关负责人
- Good　在 iCafe 中对需求统一定义优先级，并持续更新需求状态
- Excellent　需求拆分粒度大小合适，需求状态停留周期（从离开第一列到到达完成状态）小于 5 天，并持续更新需求状态

迭代管理

- Average　在 iCafe 中统一管理计划， 持续通过迭代计划进行有节奏排期，并进行项目进度　跟踪
- Good　迭代周期小于两周，平均计划完成率不低于 80%

Bug 管理

- Average　Bug 记录：在 iCafe 中统一 Bug（含线上）， 记录 Bug 内容和相关负责人
- Good　Bug 追踪：开启代码提交关联 Bug id 的开关
- Excellent　Bug 跟进：iCafe 空间中停留了半年以内的 Bug 完成率大于 80%

开发（Server/App/SDK）

本地代码检查

- Average　50% 的提交预先进行过本地检查，30% 的代码检查问题在提交前得到修复
- Good　80% 的提交预先进行过本地检查，60% 的代码检查问题在提交前得到修复
- Excellent　所有的提预先进行过本地检查，所有的代码检查问题在提交前得到修复

本地测试

- Average　使用本地代码扫描，diff 部分无严重问题
- Good　进行了两项及以上的本地自动化测试

分支规范

- Average　主干保有完整代码（分支上的修改 3 个月内须合入主干）

提交规范

- Average　清晰的主干历史（配置了“Push 仅含一个 commit”或“合入策略 =Always Merge”中的一个）

- Good 线性的主干历史（配置了“Push 仅含一个 commit” 且“合入策略 =Rebase if necessary | Fast forward only”，icode 开启关联 iCafe
- Excellent 95% 的 commit 修改不超过 400 行

关联需求

- Average 提交日志中包含有效 iCafe 卡片 id（配置了 "commit message 必须包含 iCafe 卡片 ID"）

第三方代码依赖

- Average 不能含有任何第三方不规范的依赖

代码准入（Server/App/SDK）

增量源码安全扫描

- Average 进行增量源码安全扫描，并解决所有安全漏洞

百度编码规范

- Average 进行百度编码规范检查，并解决所有检查问题

CodeReview

- Average iCode 中开启人工评审功能且禁止自评
- Good 过去 30 天的千行评论数不少于 1
- Excellent 过去 30 天的千行评论数不少于 4

增量静态代码扫描

- Average iCode 启动检查，千行高危 Bug 数低于 1
- Good 开启阻塞提交，且含 Average

云端编译

- Average 正常接入 BCLOUD，180 天内主线编译成功率大于 85%
- Good 依赖使用 Stable 分支占比不低于 70%（C++）

P0 级自动化回归测试

- Average 该环节全量覆盖率 C/C++：20%；Java、PHP 、Python： 15%
- Good 开启阻塞提交，并且有该环节，含 L0
- Excellent 该环节全量覆盖率 C/C++：30%；Java、PHP 、Python： 25%

Unit Test

- Average C/C++：增量行覆盖率 30%；Java、PHP ：增量行覆盖率 20%
- Good 开启阻塞提交，并且有 UT，含 Average

- Excellent　含 Average， C/C++: 增量行覆盖率 60%；Java、PHP ：增量行覆盖率 45%

可维护性检查

- Average　增量 MI 高于 20（事后统计不阻碍提交）
- Good　增量 MI 高于 50
- Excellent　增量 MI 高于 80

测试（Server）

全量源码安全扫描

- Average　有猫头鹰安全扫描环节，并修复所有漏洞

自动化回归测试（功能、API、schema 校验等）

- Average　自动化回归测试 +UT，全量分支覆盖率大于 40%
- Good　自动化回归测试 +UT，全量分支覆盖率大于 65%

手工测试用例管理

- Average　手工：用例在 icase 存储
- Good　手工：用例是活跃的（更新、删除、执行）

服务安全扫描

- Average　有啄木鸟安全扫描环节，并修复所有漏洞

稳定性测试

- Average　有稳定性测试环节，全量分支覆盖率大于等于 20%
- Good　有稳定性测试环节，全量分支覆盖率大于等于 40%

环境基础能力

- Average　具备可动态伸缩资源的自动化搭建环境能力，平台返回状态
- Good　具备基于线上硬件环境同质的环境能力，且数据无裁剪，平台返回状态
- Excellent　具备部署方式、硬件环境与线上一致的能力，平台返回状态

全量静态代码扫描

- Average　进行全量静描，千行高危问题 <0.4
- Good　进行全量静态代码扫描，修复所有高危问题

DIFF 测试

- Average　有 DIFF 测试环节，并有报告
- Good　DIFF 问题都得到有效跟进，如果有 DIFF 标记了原因

性能测试

- Average　覆盖至少 2 类 7 个以上指标；指标类别包括不限于：机器级别性能指标，拓扑级别性能指标，业务指标等
- Good　能产生性能白盒数据或指标
- Excellent　提供性能分析报告，给出明确的结论和过程数据，并有 xray 的系统分析报告

异常测试

- Average　有异常测试环节，且异常测试场景大于 1 个
- Good　异常测试场景大于 3 个（接口、数据、网络、硬件等）

压力测试

- Average　能够找到系统 / 模块的极限吞吐量
- Good　Average 基础上，能够获取影响极限吞吐量的核心因素和关系，指导模块或系统优化

测试（App/SDK）

全量源码安全扫描

- Average　有猫头鹰安全扫描环节，且发布前修复所有漏洞

系统兼容性测试

- Average　有兼容性测试环节，覆盖机型或版本 10%
- Good　有兼容性测试环节，覆盖机型或版本 40%

自动化回归测试（含 UT）

- Average　自动化回归，全量分支覆盖率大于 10%
- Good　自动化回归，全量分支覆盖率大于 20%

稳定性测试

- Average　稳定性执行时间超过 30min，平均每小时覆盖页面数（去重）80+
- Good　稳定性执行时间超过 30min，平均每小时覆盖页面数（去重）150+

手工测试用例管理

- Average　用例在 icase 存储
- Good　用例是活跃的（更新、删除、执行）

包安全扫描

- Average　有安全扫描环节，修复全部漏洞

全量静态代码扫描

- Average　进行全量静描，千行高危问题 <0.4
- Good　进行全量静态代码扫描，修复所有高危问题

性能测试

- Average　有能力判断至少 2 项测试结果异常（内存、CPU、耗电、流量等）
- Good　有能力判断至少 4 项测试结果异常（内存、CPU、耗电、流量等）

性能评估

- Average　具备自动化能力，至少返回 8 项不同场景指标
- Good　有竞品分析报告

UI 兼容性测试

- Average　覆盖的机型数 * 页面数（去重）大于 100，并且有 2 种以上 UI 检测
- Good　覆盖的机型数 * 页面数（去重）大于 200，并且有 4 种以上 UI 检测

异常测试

- Average　有异常测试环节，且异常测试场景大于 1 个
- Good　异常测试场景大于 3 个（接口、数据、网络、硬件、低电量等）

隐私合规测试

- Average　检测并修复产品中存在隐私不合规问题

上线 & 验收 (Server)

包管理规范

- Average　使用 Agile 产品库进行上线
- ≠ Good　产品库含 PaaS 平台标准描述文件

自动检查能力

- Average　P0 Case 检查 + 核心模块监控检查（FATAL，CORE）+ 核心可用性检查
- Good　核心业务指标检查 + 性能检查大于 3 项
- Excellent　性能退化检查 + 模块级 Case 检查 + 端上打点数据检查 + B 端客户可用性检查

分级部署能力

- Average　使用了标准的上线变更平台且且模块具备按机房分级发布能力
- Good　使用公共的 PaaS 平台部署，具备沙盒环境提供能力
- Excellent　具备按流量比率进行灰度发布能力

容量评估

- Average　执行入口级容量测试，找到系统总吞吐量
- Good　在 Average 的基础上，给出单层 service 的容量
- Excellent　在 Average 的基础上，给出单层 service 的容量

灰度（App/SDK）

灰度发布基础能力

- Average　使用正规的包打包平台和灰度平台进行灰度
- Good　覆盖人数或覆盖系统占比 10%

灰度发布控制能力

- Average　具备灰度发布能力（灵活比例、渠道）
- Good　具备灰度止损能力

灰度数据评估能力

- Average　发版前新增 crash、anr 必须修复
- Good　具备灰度分析报告，指标数不少于 6 项
- Excellent　用户反馈处理率 80%

发版（App）

规范的包版本管理

- Average　使用包管理平台进行产品包管理
- Good　可以方便的获取历史包

规范 APP 自动出包

- Average　在包管理平台中制作渠道包
- Good　在流水线中全自动制作渠道包

可控的发布过程

- Average　具备控制渠道的止损能力
- Good　具备线上热修复能力

交付（SDK）

包版本管理

- Average　使用 Agile 产品库进行交付物版本管理

流水线 & 自动化（Server/App/SDK）

开发

- Average　构建时长 <2h，红灯修复时间 <8h

- Good 构建时长 <1h，红灯修复时间 <4h
- Excellent 构建时长 <30min，红灯修复时间 <1h

代码准入

- Average 构建时长 <30min，构建成功率 >85%
- Good 构建时长 <15min，构建成功率 >90%
- Excellent 构建时长 <5min，构建成功率 >95%

测试

- Average 构建成功率 >75%，主线红灯修复时间 <8h
- Good 构建成功率 >85%，主线红灯修复时间 <4h
- Excellent 构建成功率 >95%，主线红灯修复时间 < 1h

上线

- Average 构建时长 <2h
- Good 构建时长 <1h
- Excellent 构建时长 <30min

DevOps 实施效果

研发交付周期

- 第一次开发为首次发起代码评审的时间。单需求平均交付周期，需求『第一次开发』的时间 到『已发布』的间隔时间分位值

线上质量

- 单位时间 P0&P1 线上问题数

问题平均修复时长

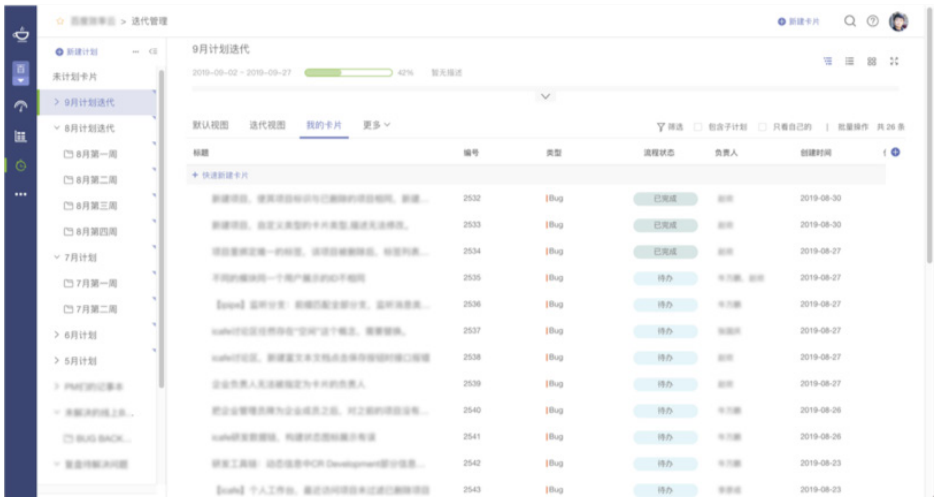
- P0&P1 问题从【发生】到【关闭】的平均时长，分位值

百度软件工程标准的实施

使用百度效率云实施工程标准

上述的工程标准有助于团队了解工程全貌，自主确定工程改进目标，自助改进工程能力。每一个工程实践都有对应的工程工具平台支持，团队只需在研发工具中选择对应的工具插件，即可获得相应的工程能力。百度研发工具平台——百度效率云是支撑百度工程标准快速落地实践的唯一平台。百度效率云包含三个主要工具：项目管理平台 iCafe、代码管理平台 iCode 和持续交付平台 iPipe。

例如在工程标准的“需求”阶段中，需求管理、迭代管理和 Bug 管理实践，均可以使用百度敏捷项目管理工具 iCafe 来实现。

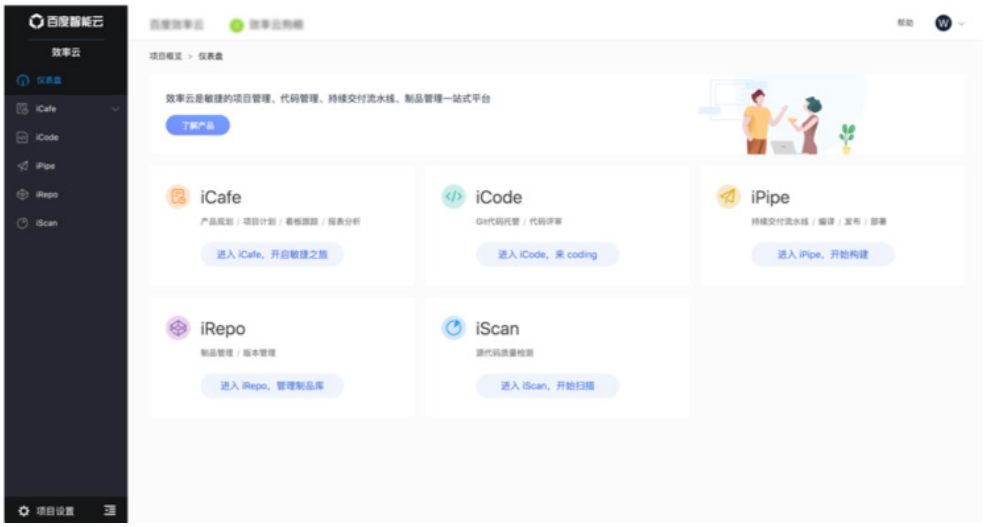


在“代码准入”阶段中的若干工程实践，都可以从百度代码管理工具 iCode 中进行选择。

在“测试”和“上线验证”阶段中的工程实践，则可以从百度流水线平台 iPipe 中选择相应插件，下图为百度研发流水线 iPipe 的部分插件：

配置完成 iPipe 流水线以后，每当代码提交、或代码合入时，流水线就会自动执行测试、验证等任务。在达到工程标准要求的同时，显著提高了研发效率和质量。

百度效率云 DevOps 服务不但对百度内部提供服务，也在百度智能云上对外开放。为广大软件研发团队提供与百度内部一样强有力的工程能力保障，帮助团队提高效率和质量。

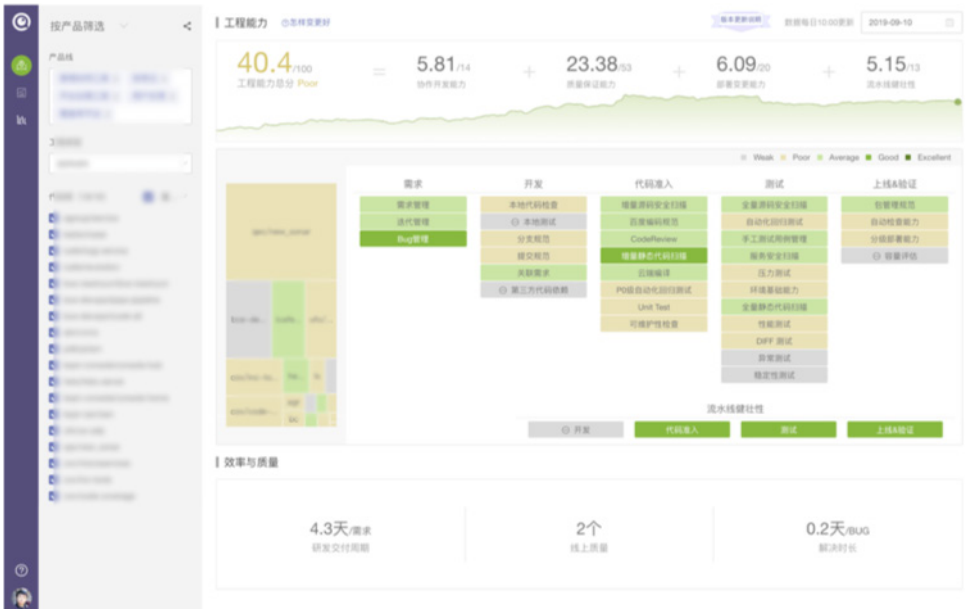


https://cloud.baidu.com/product/xly.html

验证方式

团队在使用工程工具、按照工程标准进行工程实践的过程中，这些工程工具平台能够按照统一规范回传数据，进而可以分析计算每个团队在每个工程实践上已经达到的级别。为此我们用研发工具平台产生的研发现场数据，建立了百度研发数据仓库，并基于这些工程数据，开发了工程能力地图可视化的产品，方便各个团队了解自身的工程能力，验证工程改进的效果。

工程标准从 V1.0 升级到 V2.0 的同时，“工程能力地图”可视化工具也进行了同步升级。后端的自动化计算规则与新的标准保持一致，同时增加了结果指标的显示。



实施效果

我们持续滚动对百度所有团队的研发数据进行定量分析。分析结果表明，团队采用的工程实践数量越多，其开发周期越短；工程实践做得程度越深入的团队，其开发周期也越短。同时研究了开发团队人数对上述结果的进一步影响，发现团队人数越多，实施工程实践对缩短开发周期的作用就越大。以上分析结论进一步佐证了优秀的软件工程实践能够缩短开发周期。

未来与展望

本白皮书介绍了 2019 年百度工程的标准和工程实践的升级情况，增加了 DevOps 结果指标的定义和度量，对外发布了支撑百度功能标准落地实施的 DevOps 工具平台——百度效率云。

对于一个科技公司来说，保持并不断提升工程能力是最重要的。百度一直坚持培养工程师的工程素养，建设高效的工程工具，持续推动工程能力提升，Relentless Pursuit of Engineering Excellence，永无止境的追求工程卓越。

我们作为软件工程的实践者，希望为同行或研究机构提供一个参考样本，促进交流，共同提高。百度工程标准和工程工具会继续随着公司内外工程技术的不断革新而持续升级。百度软件工程团队希望及众家之所长，合众人之智慧，以实践为检验标准，促工程能力打造科技产品，用科技让复杂的世界更简单。

不懈追求　工程卓越