ASD工程效能分享交流

ASD | 陈潜

00 工程效能导入

定义、目标、意义

- **工程效能 = 干更有价值的事 + 干得更快更好更省**
- **工程效能是技术企业的核心竞争力，是公司重要的组织能力**

研发投入

需求响应 需求分析 等待 需求评审 设计 开发 测试 运维

新建需求 响应需求 确定排期 迭代开始 启动设计 进入开发 提测 上线

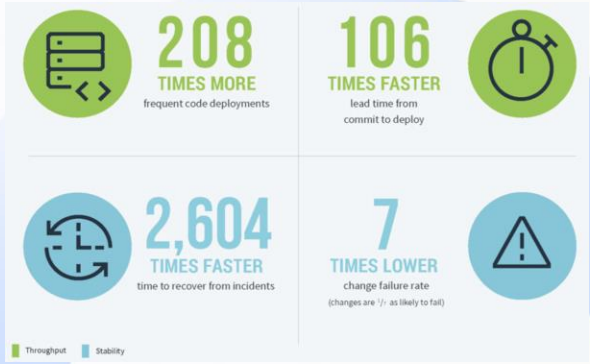
业务产出

工程效能聚焦在让人力投入更高效：

- 人力投入分布：研发投入+非研发投入占比
- 提升非研发活动效率：
  - 降低值班答疑、开会效率、知识沉淀、新人快速培养...
- 高效端到端研发工具链：
  - 需求如何拆解,拆多大？PM需求有效性、研发交付SLA
  - 研发流程、开发环境、框架选型，分支规范
  - 测试Case、框架，流水线建设
  - 分级发布，监控、日志、预案

不同企业、团队工程效能差异巨大

- 阿里211：85% 的需求 2 周内交付完成，1 周内开发完成；创建变更后 1 小时完成发布
- Feed：50%+的需求天级交付(开发->上线)



百度智能汽车事业部（ASD）

打造『有量产且领先』的专业、新型、本土的汽车智能化Tier1

智驾、智舱、智图

质量保证（Q）

效率提升的同时，
保证质量稳定

成本控制（C）

减少人工干预，
降低人工工作量

快速交付（D）

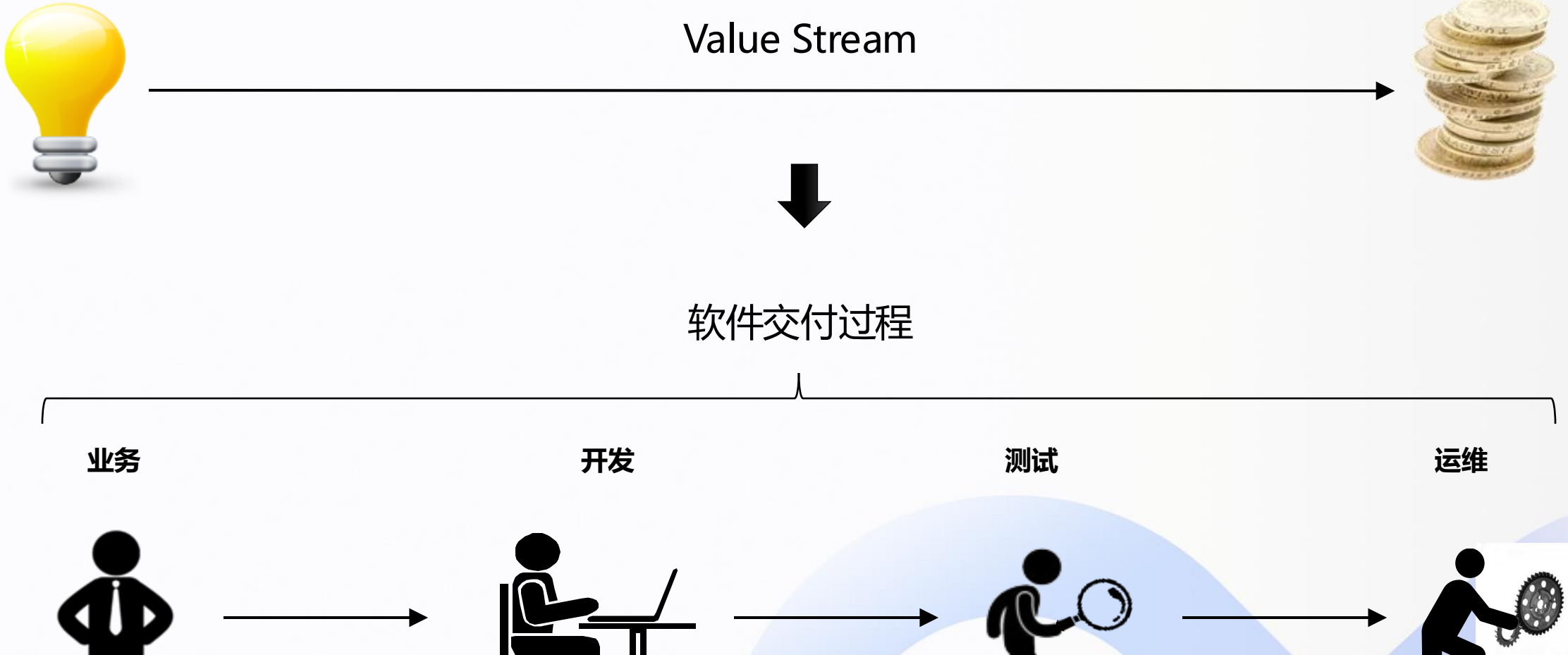
从需求到交付端到
端交付周期缩短

01 完善合理的度量体系

『诊断问题+指导提效+验收成果』

01 软件交付过程

典型的软件交付过程



01 软件交付过程

核心问题分析：周期长、效率低

人员、流程、技术被「墙」阻断，Throw it over the wall...

业务

墙

开发

墙

测试

墙

运维

- 需求以文档传递，缺乏沟通
- 需求不易理解，且经常变更

- 测试反馈周期长，测试占研发比重大
- 自动化测试程度低，质量把控不严格

- 运维排期紧张，上线存在等待
- 手工部署繁琐，易出现人为错误

效率低，存在浪费

反馈慢，质量保证不全

故障多，操作耗时长

传统软件研发和交付过程无法满足业务快速、稳定交付的要求

01 软件交付过程

『我们的首要任务是尽早持续交付有价值的软件并让客户满意』

需求拆分，小步快跑

尽早的、持续的测试

开发到运维流程打通

Scrum

持续集成

DevOps

端到端交付过程优化，在保证质量的前提下提升效率

持续交付

关键点

小批量持续交付 各环节效率提升 减少环节间等待和浪费 降低协作成本 优化工具效能

02

敏捷迭代

常见的敏捷实践

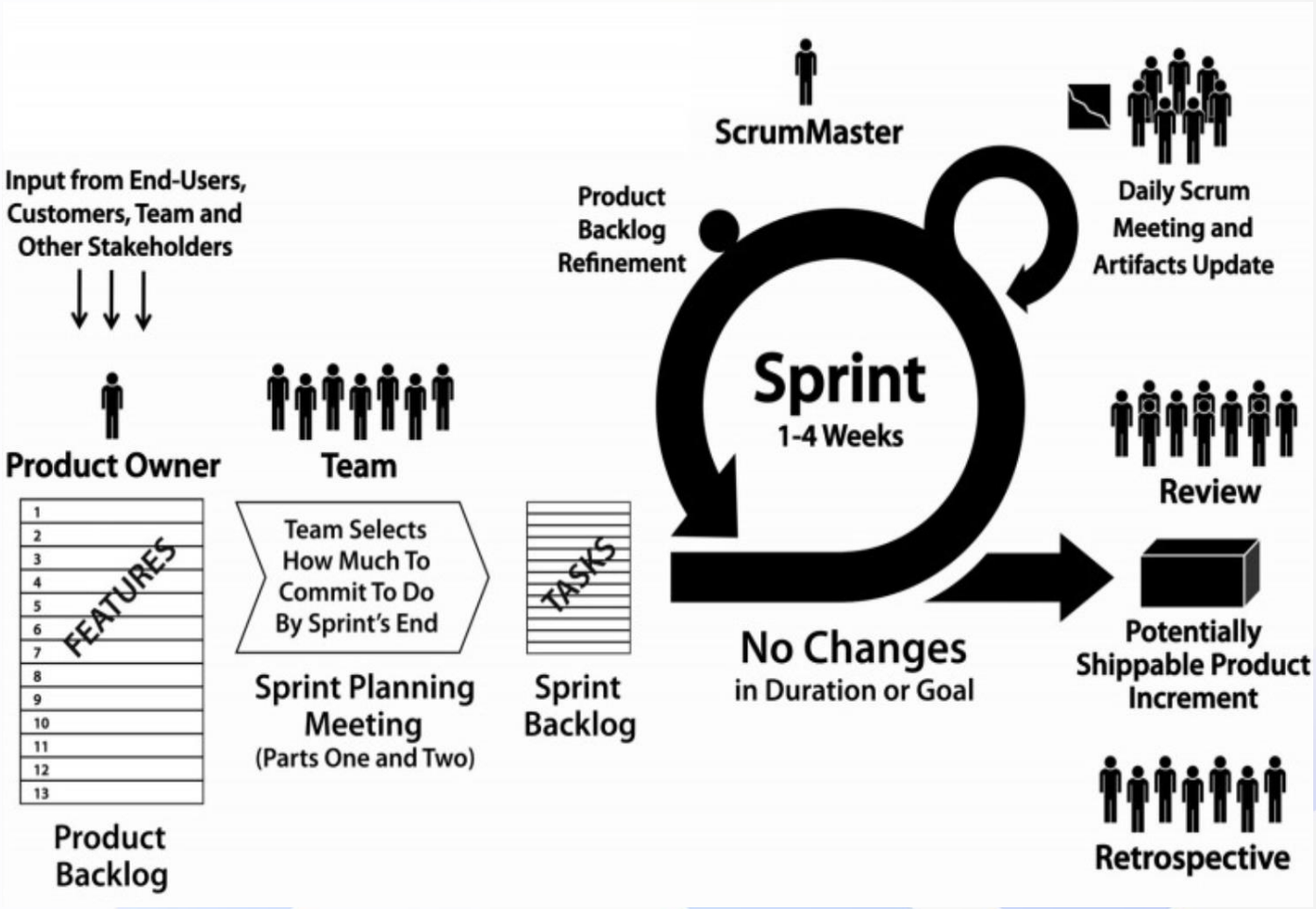
Baidu 百度 | apollo

Scrum	• 迭代式增量软件开发过程，用于敏捷软件开发。 • 迭代前需求排优先级和细化，迭代中每天站会同步进展，迭代后交付与复盘
持续集成	• 软件开发和运维实践 • 每日自动构建和发布
看板管理	• 项目进度跟踪，RD/PM/UE/QA交流沟通，了解彼此进展和问题。 • 状态定义，色区分卡，内容、姓名、角色等标识，燃尽图
每日站会	• 每天15分钟，固定时间 ，同一时间同一地点互相沟通和分享工作 • 分享内容： 昨天做了什么？ 今天会做什么？遇到的困难？

02 敏捷迭代

Scrum框架：角色和流程

- **产品负责人 (Product Owner)**
  - 对团队对外交付的价值负责
  - 定义需求
  - 定义需求的优先级
  - 定义需求的验收标准
  - 定义产品发布内容与日期
- **敏捷教练 (Scrum Master)**
  - 帮助团队遵循Scrum 框架并持续改进
  - 促进团队的工作，保护团队不受打扰
  - 帮助团队熟悉与掌握 Scrum 价值观与框架
  - 帮助团队排除影响生产力的障碍
- **团队 (Scrum Team)**
  - Scrum team 对交付成果负责。
  - 跨职能部门
  - 自组织式的团队
  - 小而美

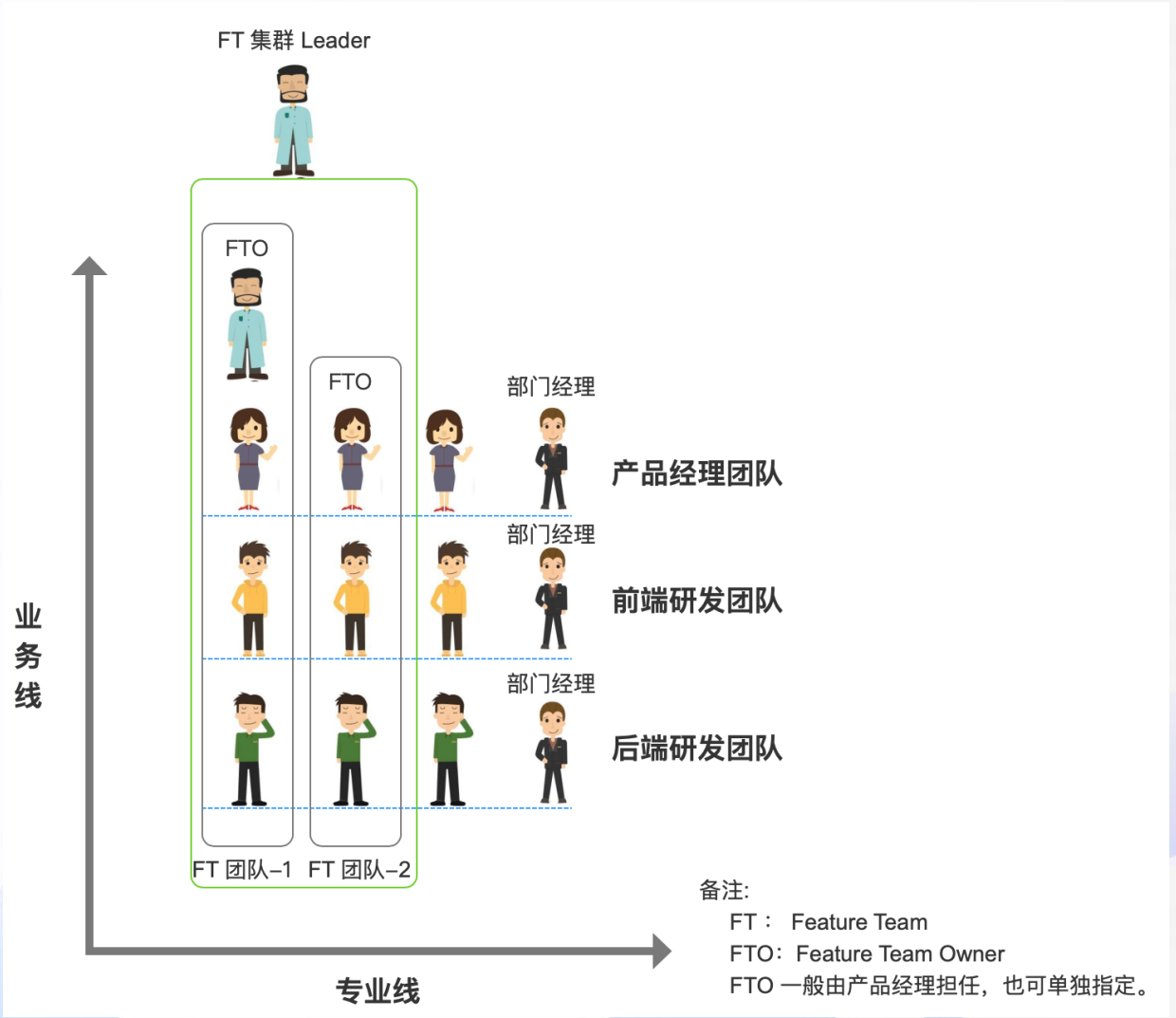


02 敏捷迭代

按产品组建跨角色的敏捷组织

- **Why**
  - 以用户价值为中心交付完整的功能
  - 减少依赖，团队成员共同为业务目标负责
  - 简化的计划
  - 高效的沟通
  - 加快产品上市时间

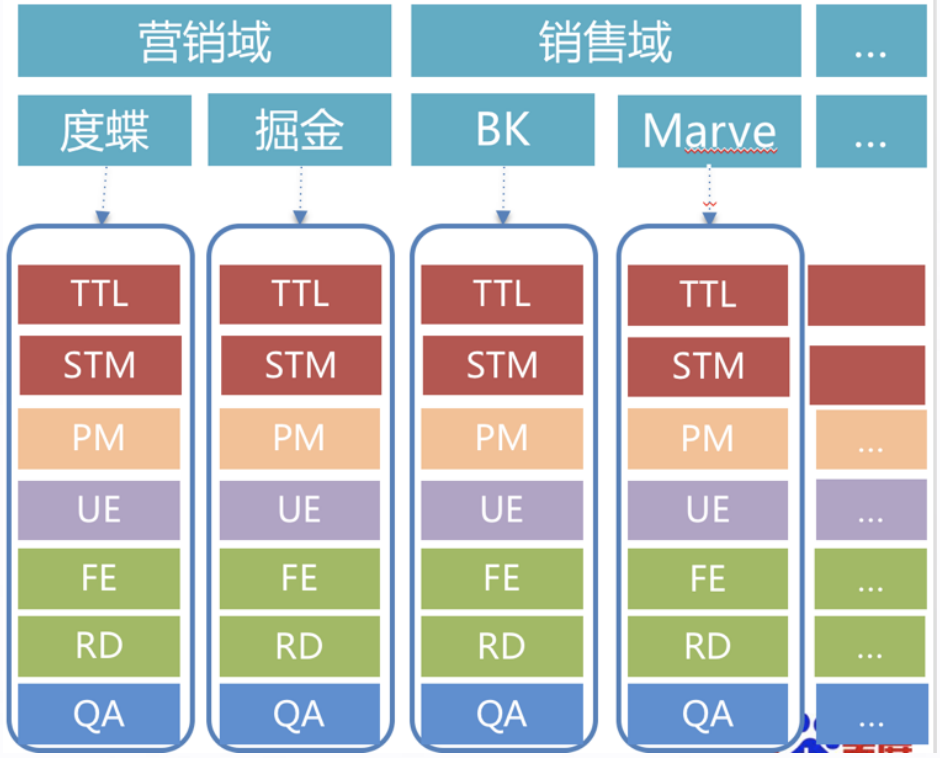
- **人员建议**
  - 5~11人，虚拟团队
  - 覆盖端到端实现产品功能全角色
  - 尽量避免一个人在两个团队中同时肩负职责

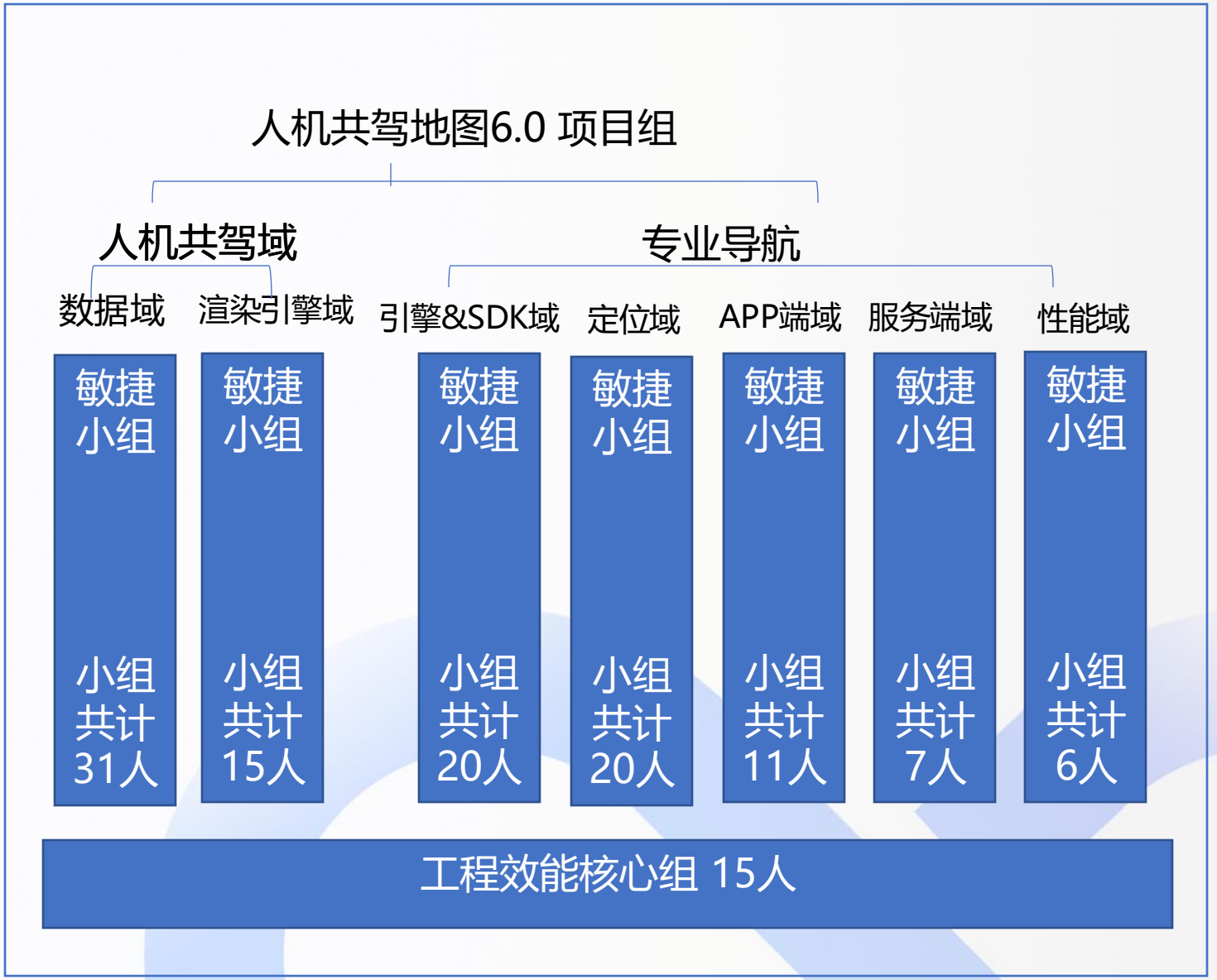


02 敏捷迭代

敏捷组织建立

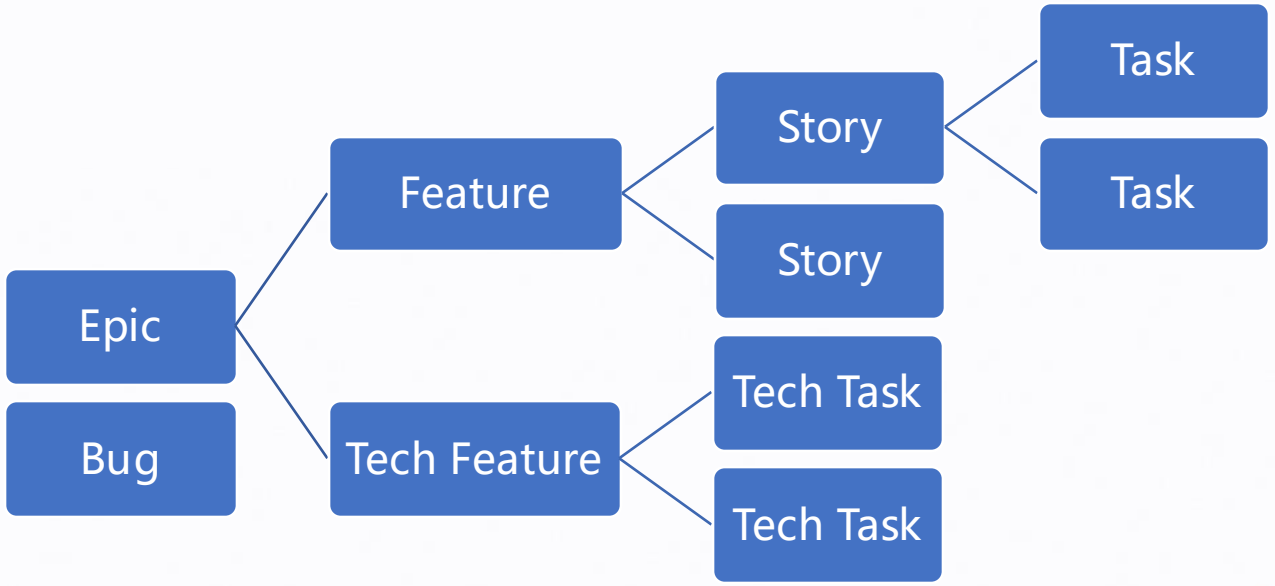
- 改造后组织结构-以业务划分





02 敏捷迭代

需求拆分：聚焦用户价值，迭代增量发布，快速反馈闭环



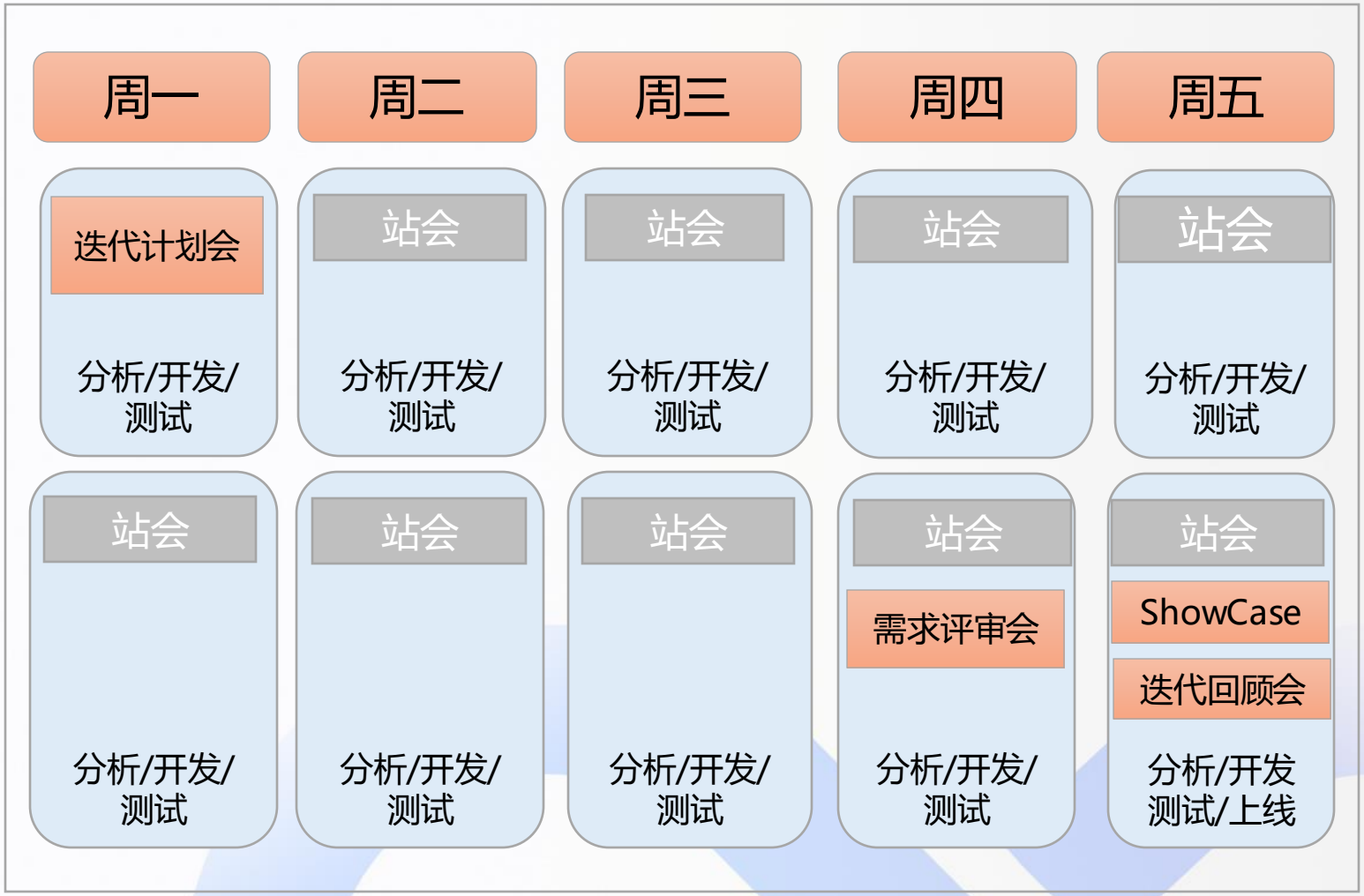
- **Epic**：系统类型，PM负责，根据各产品季度OKR拆分的史诗级事项，跨季度的Epic建议拆分，一个季度内可完成
- **Feature**：系统类型，PM负责，根据Epic拆分的用户可感知的，体现端到端完整价值的业务需求，可独立交付
- **Story**：系统类型，PM负责，根据Feature拆分的用户故事，可独立测试的系统/模块增量，一个迭代内可完成
- **Task**：系统类型，RD负责，根据Story拆分的Task，多人负责的Story需要拆分Task，Task必须有唯一负责人，提交代码必须绑定叶子节点的卡片
- **Tech Feature**：系统类型，RD负责，技术需求，如性能、稳定性、工程能力等方面的需求
- **Tech Task**：系统类型，RD负责，根据Tech Feature拆分的技术任务， Tech Task必须有唯一负责人，提交代码必须绑定叶子节点的卡片
- **Bug**：系统类型，QA负责

02 敏捷迭代

双周迭代日历

迭代日历

- 迭代的节奏感让团队能够以稳定的步调工作，使单调而必要的活动成为习惯
- 因为每个人都知道这些活动在什么时候进行，所以显著降低了安排迭代会议所需要的成本
- 可以为每个人的日程表发出一个重复发生的事件
- 如果多个团队在做同一个项目，让所有的团队都使用同样的冲刺节奏
- 明确关键节点，如需求评审、提测时间、产品 show case 时间、外部团队集成时间等



02 敏捷迭代

双周迭代，高效站会信息同步，流程工具化

- **高效站会，及时识别问题和风险**
  1. 小团队建议全角色参会，大团队/跨团队站会可以分层、分方向
  2. **严格控制时间15min，**使用看板，每个 RD 回答『我昨天完成了什么？我今天要做什么？我被什么东西阻碍了吗？』

错误的站会

- **站会不是一个汇报会。**传统的汇报会很呆板，大家基本没有交流，汇报的对象往往是老板或者组长，气氛不够放松，也不够开放。
- **站会不是知识分享会。**当我们解决一个新问题或使用一个新技术时，就特别期望能与队友分享，它要占用很多时间，而且也许只有部分人感兴趣。所以站会不适合做知识分享。如果需要分享，团队可以定期按需求组织此类活动。
- **站会不是细节讨论会。**尤其是出现问题时，团队非常容易引发细节讨论，同样也要占用较长时间，我们可以在站会后组织一个跟踪会议，请相关人员与会即可。

站会的技巧

- **坚持使用令牌**，促进每个人按顺序主动讲，不需总点名，也不会乱。SM最后讲，先让大家讲；
- **大家站成半圆形**，每个人都能一排看到。不要站成两排，后面的人就会开小差。保持好的阵型，能提高站会效率；
- 避免站会变成给SM的汇报会，让**每个人多思考**
- 对于讲话较少的同学，你可以采用一问一答的方式，促进多说话和讲问题，**及早暴露风险**
- 每个人围绕自己的**需求卡片**的进度、质量和障碍去讲
- SM做好整体**控场**
- 每个人负责移动自己的任务卡，并及时注意完成的定义(**DoD**)；

02 敏捷迭代

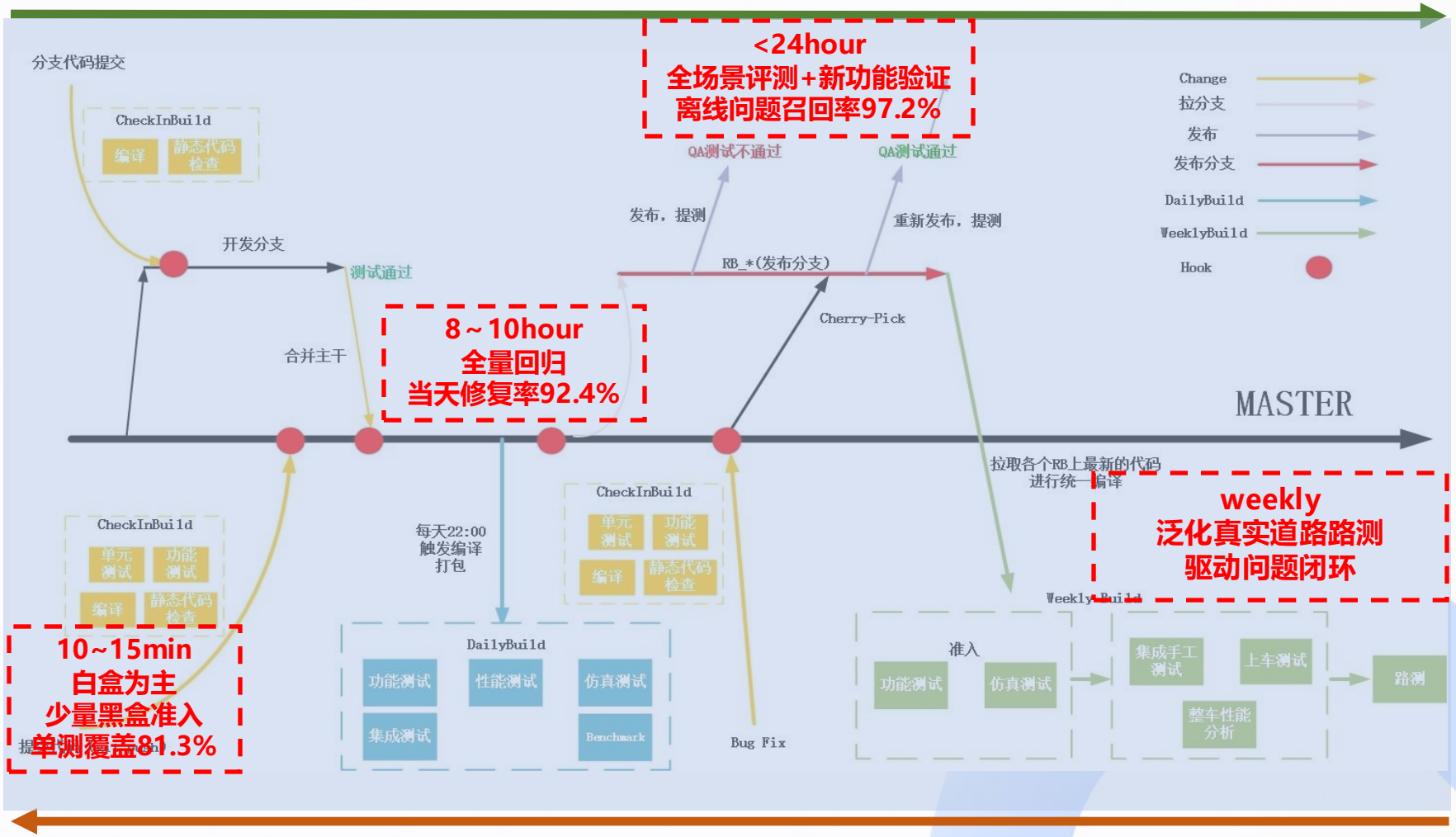
及时复盘并监督落地

- **时机：**
  - 迭代回顾：迭代周期的最后一天
  - 小项目复盘：项目结束
  - 大项目复盘：阶段结束、项目结束
- **执行要点：**
  - **迭代回顾：**做的好的地方、需要改进的地方、下一步 to do
  - **项目复盘：**回顾目标->分析结果->评估结果->总结规律
  - 当问题过多时，选择高优 TOP3
  - 要形成明确的Action，确定跟进者和完成时间，确保闭环
  - 常见问题：需求澄清不够、不断变更等

03 高效反馈闭环

15min提交级验证，天级集成验证，天+级上路测试，双周级发布交付

外循环：层层递进，环环相扣，高效迭代



内反馈：质量前置，降低修复成本

高效稳定的构建系统

- 分布式编译系统
- 多架构支撑
- 测试服务
- 智能化提效

全面高效的测试体系

- 高效统一测试框架
- 仿真模拟系统
- 数据驱动测试与评价

合理的流程规范

- 三方库&模块管理规范
- 测试规范

04 实践分享

痛点1：多车厂交付下，多架构的C++构建效率提升

编译优化

18年至今IDG编译集群任务统计

- **分布式：18年完成探路者BCLOUD迁移**
  21年智驾、智图业务
  22年ANP2-AVP改造

IDG缓存命中率92.4%，全集成时间14min

多架构分布式构建（4M解决方案）

搭建公司内部首条交叉编译分布式构建系统，多架构构建耗时<10min

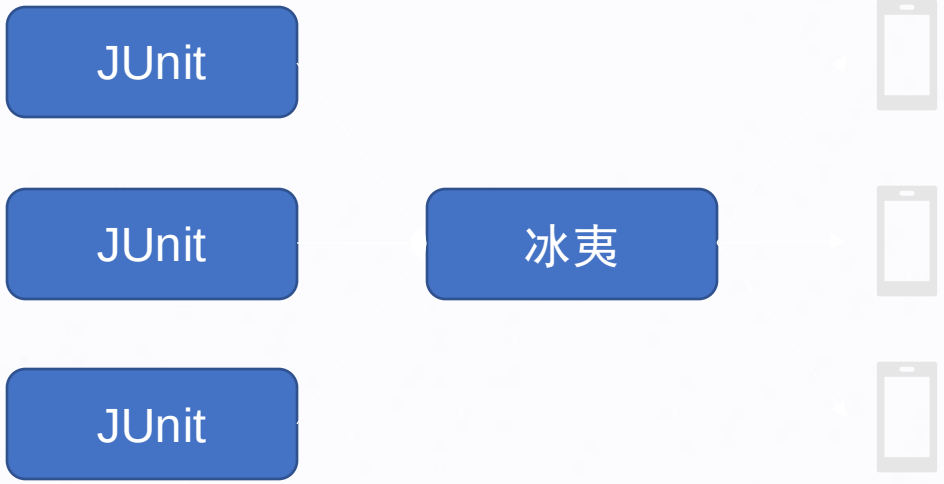
- 探路者DCU（aarch64）构建耗时4h->4min，智驾45min->5min（四喜&三鲜）
- 车联网6系（安卓：aarch64+armeabi-v7a+armeabi）构建耗时100min->5min
- 高精引擎&自定位（4+）构建耗时23min->9min

业务支持	探路者、车联网、高精自定位、ANP、交通、ACU	基础库/第三库支持	
	多平台业务上线	编译技术专家团队支持	
核心能力	4M配套系统能力	多依赖库管理	多平台产出功能
		ARM Docker构建	自定义ARM编译机
		本地编译 bcloud local	分布式编译 bcloud build
		原生编译	交叉编译
		bcloud 服务兼容适配多架构、多平台	
多工具链	标准编译工具链	车厂定制化编译工具链	
多系统	社区系统 Centos/Ubuntu/安卓	国产化系统 kylin/UOS/..	
多架构	x86_64	aarch64	arm	sw64	mips	...

04 实践分享

痛点2：安卓项目车机资源不够，无法坐到checkin级别的验证

- 基于TPG 冰夷平台及百度云手机实现SDK API测试任务并发、Case并行



设备sn	压测进程	平台类型	测试版本	创建人	icafeid	状态	创建时间	操作
Android10_yun_phone_1	normalandr_MapAutoMirrorActivity_test1	Android	6.0.6814.e815a5c718824	wangshu08	0	已结束	2022-06-16:39:37	详情 停止 重启 日志
Android10_yun_phone_10	normalandr_MapAutoMirrorActivity_test1	Android	6.0.6814.e815a5c718824	wangshu08	0	已结束	2022-06-16:39:16	详情 停止 重启 日志

04 实践分享

痛点3：高频迭代情境下，测试如何满足迭代质量与效率的需求

按照每个阶段效率与质量要求结合测试能力的当前特性进行合理部署

质量前置

➢ 纵向阶段前置

- **案例：性能测试服务化（1）**

问题：环境搭建与问题分析成本高

迁移路径：Daily阶段->CheckIn阶段->Local阶段

技术路径：PiL动态资源管理与调度、性能测试精准化、任务服务化与平台化

效果：线上性能问题快速收敛，性能作为基本的准入

➢ 横向分层前置

- **案例：7*24稳定性测试（2）**

问题：无法支持长时间不间断的闭环

迁移路径：道路测试->HiL->PiL

技术路径：全链路仿真、长时间连续场景数据->时空循环连续场景（GNSS-MOCK）

效果：较低成本实现常稳测试，实现Daily级别的稳定性问题拦截

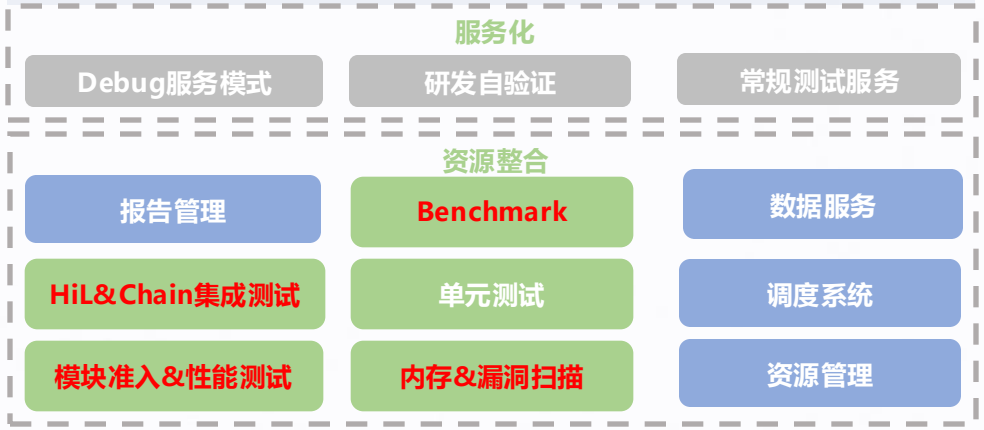
04 实践分享

智能化、服务化构建测试中台系统，大幅降低测试成本与测试冗余

Baidu 百度 | apollo

统一服务化平台

➢ **发布新服务化架构Biling，聚合测试资源与能力**
对外提供测试服务，支撑研发自测，Checkin通过率**提升21.4%**
对内提供动态资源池，串联流水线，部分实现无人值守能力



支撑3538个Daily，513个RB
支撑82124次CheckIn，24715个模块测试任务
支持2424个集成性能测试，评估服务715次

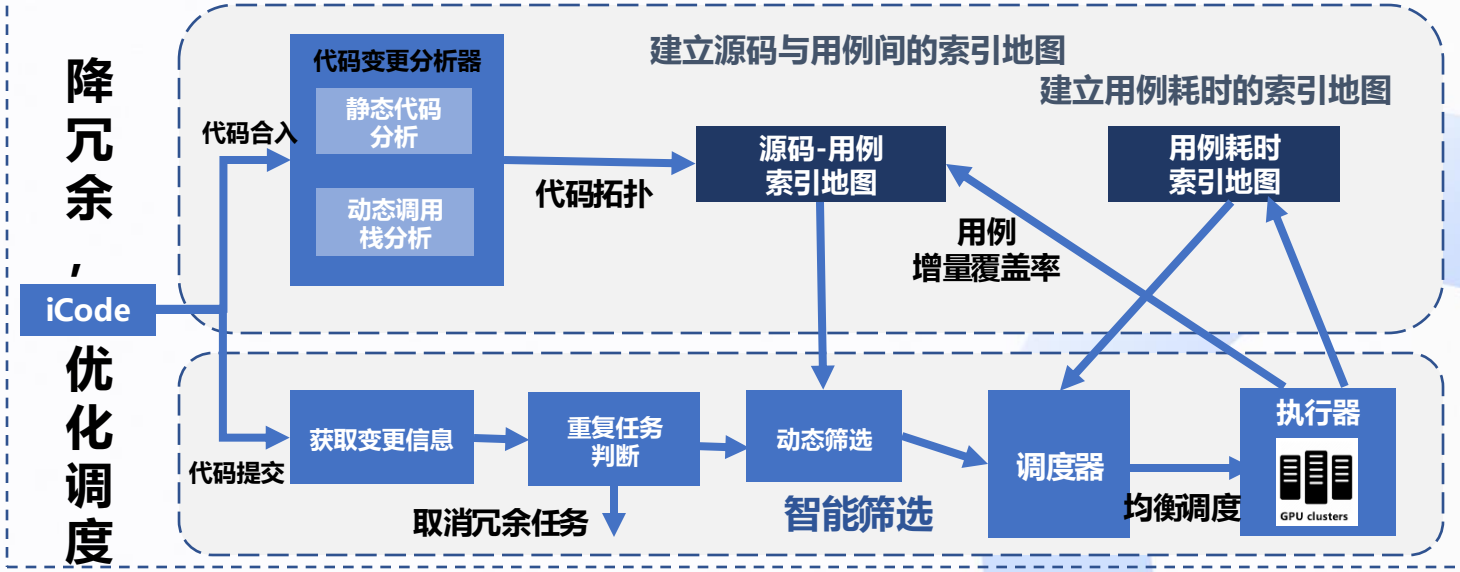
整体任务通过率81.5%，平台类异常占比0.43%

打磨测试标准

精细化性能测试
端到端耗时标准差<5ms

智能化构建策略

降冗余，优化调度



智能调度
根据用例历史执行时间，**动态分片调度**，实现分布式环境下的负载均衡

智能筛选
识别patchset，**实现流水线级的冗余过滤**
通过源码用例索引地图，进行**用例级的动态筛选**，实现精准覆盖

完成探路者全核心模块（34）智能化构建落地
构建效率提207%

谢 谢 观 看!