agile tour 2021

如何做好软件交付过程改进？先做好“体检”

主讲人：董越（流水先生）

主办方：禅道 Zen&Tao

自我介绍

董越

DevOps咨询师

《软件交付通识》作者，业界资深DevOps专家，前阿里巴巴研发效能事业部架构师，Certified DevOps Enterprise Coach


群众意见

目录
contents

PART 1

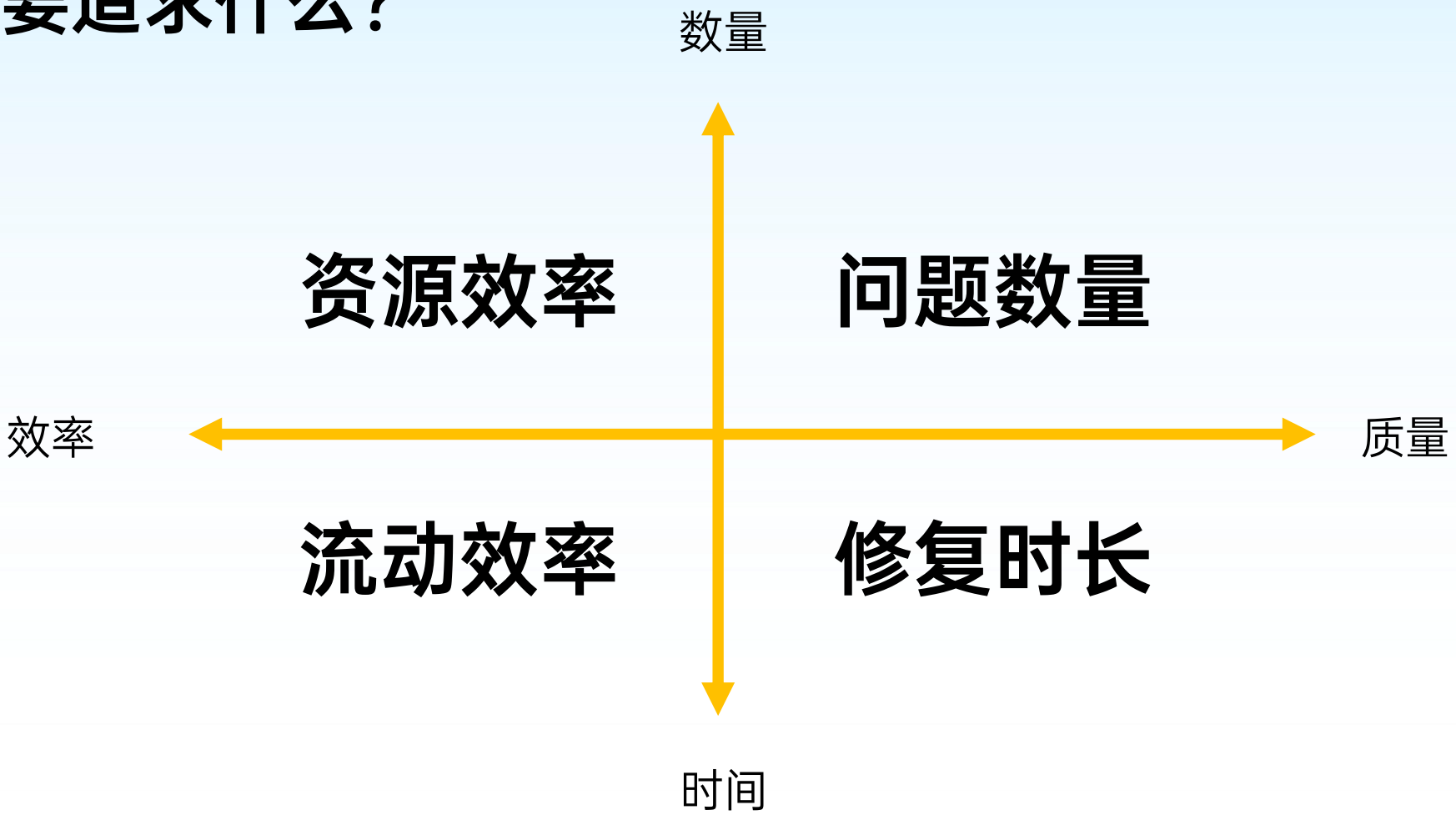
我们要追求什么

聚焦软件交付过程

软件交付过程（Software Delivery Process）是指编程序改代码之后的一系列活动，直到软件发布给用户使用。

在大图中的位置

我们要追求什么？



PART 2

了解项目和团队总体情况

业务和系统的主要特征

业务

这个系统/模块的用途？
ToC、ToB还是企业内部使用？

规模

软件规模、研发团队规模？
用户规模、部署规模？
耦合性如何？

形态

服务端：SaaS？企业内部部署？
客户端：浏览器？移动端？

质量

质量要求高不高？
好实现吗？

管理实践

需求层级

各级需求的名称?
如何定义该级别?

发布节奏

发布频率?
与迭代的关系?

计划管理

Scrum迭代?
精益看板墙?

组织架构

开发/测试/运维团队?
工具/流程/改进团队?

从源代码到程序运行

源代码

编程语言?
代码库类型和数量?

部署运行

通过脚本部署?
通过配置管理工具部署?
Docker+Kubernetes?

构建

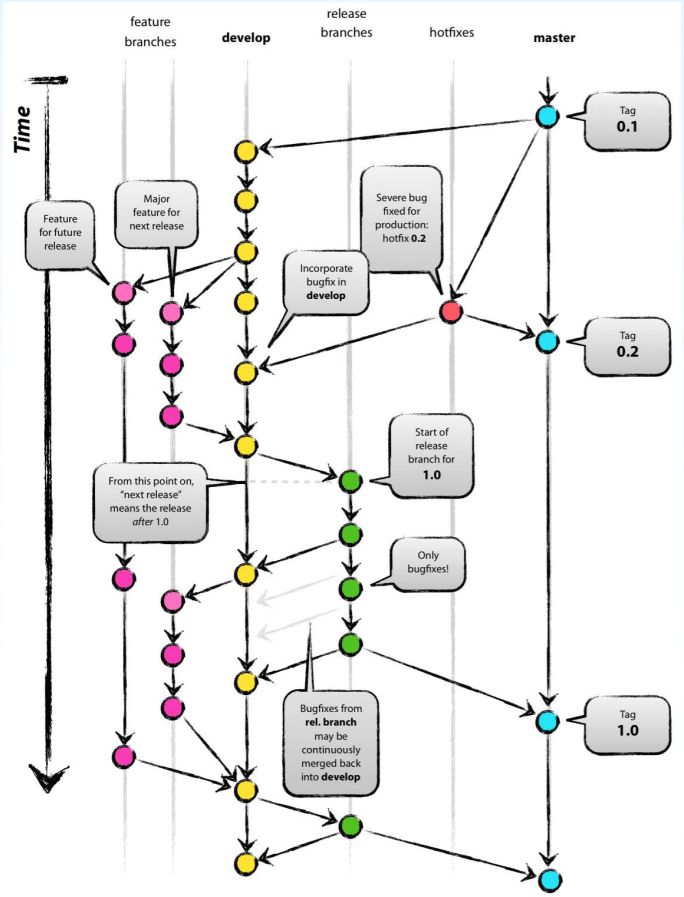
安装包类型和数量?
静态库情况?

运行环境

有个人开发环境?
有哪些测试环境?
有生产环境?

分支策略

- 总的分支模式
- 线上最新版本分支
- 集成/发布分支
- 特性分支



PART 3 沿价值流进行梳理

软件交付过程六阶段

- 代码改动累积
- 代码改动提交
- 特性改动累积
- 特性改动提交
- 集成
- 发布

如何梳理各阶段中的活动

- 按顺序梳理该阶段串行和并行的各个活动，包括可选的
- 每个活动取哪个分支的代码，在哪个环境执行
- 每个活动的执行频率，触发时机
- 每个活动的时长，等待和返工的情况
- 每个活动的相关角色，其归属团队
- 每个活动自动化的情况，以及是否纳入自动化流程中

代码改动累积

- **执行时机**
  - 实时语法报错、代码扫描
  - 随时构建、单元测试、代码扫描
  - 随时端到端的人工自测试
- **执行效率**
  - 使用IDE

代码改动提交

- **执行时机**
  - 代码提交频率较高
  - 提交在逻辑上完整地完成了一小块改动
- **执行效率**
  - 代码改动提交有一定的内容说明
  - 代码改动提交关联到相应工作项
- **执行效果**
  - 已提交改动的构建、单元测试、代码扫描方面的问题不多
  - 已提交改动所包含的运行时的低级错误不多

特性改动累积（1）

- **执行时机**
  - 代码改动提交触发流水线执行构建、单元测试、代码扫描
- **执行效果**
  - 分析逃逸缺陷
- **执行效率**
  - 无需为每条特性分支单独配置流水线
  - 统一的流水线服务

特性改动累积（2）

- **问题处理效率**
  - 提交触发的流水线执行失败时，通知代码改动者
  - 使用即时通讯工具通知
  - 自动记录流水线配置的修改历史
- **避免引入问题**
  - 流水线服务稳定性

特性改动提交（1）

- **执行时机**
  - 特性开发时间较短，特性提交时代码合并冲突较少
  - 人工进行该特性的自测试
  - 质量门禁包括对构建、单元测试、代码扫描、代码评审的质量要求
- **执行效果**
  - 已提交特性的构建、单元测试、代码扫描方面的问题不多
  - 已提交特性所包含的运行时的低级错误不多
  - 分析逃逸缺陷

特性改动提交（2）

- **执行效率**
  - 提交门禁的代码评审耗时较短
  - 提交门禁的构建和自动化测试耗时较短
  - 特性分支和其合并请求关联到相应工作项
- **问题处理效率**
  - 自动通知代码评审人或特性作者进行进一步的操作。
  - 特性摘除方法合理，操作方便。

集成（1）

- **执行时机**
  - 比较持续地接收特性提交
  - 代码改动提交触发流水线执行构建、单元测试、代码扫描、自动化冒烟测试
  - 随时进行新特性相关的人工测试和自动化测试
  - 适当的制品晋级策略
  - 不同计划发布版本的开发-交付过程适当交叠
- **执行效率**
  - 使用流水线自动串接起集成过程中所有活动，如部署、自动化测试等
  - 维护测试或发布版本所对应的特性列表
  - 开发团队可便捷地自主完成构建和部署流水线上的配置

集成（2）

- **问题处理效率**
  - 较短的红灯修复时长
  - 较短的缺陷修复时长
  - 流水线执行后适当通知
- **避免引入问题**
  - 流水线、工作项、看板墙等，对本开发团队以及所有相关人员可见
  - 每个人使用自己的账号而不是公共账号操作流水线

发布（1）

- **执行时机**
  - 发布频率高
  - 不进行人工的全量回归测试
  - 发布或更早的质量门禁包含对人工&自动化测试的质量要求
  - 不限制发布日期，任意日期都可以发布
  - 经常以开发团队甚至微服务为单位集成和发布
  - 妥善处理不同微服务或多种变更内容间的版本兼容性造成的部署顺序约束
  - 妥善安排静态库相关的集成和发布

发布（2）

- **执行效果**
  - 上线后暴露出的故障和缺陷的数量不多
  - 上线过程比较顺畅，发布成功率比较高
  - 分析集成和发布过程中遇到的问题和发现的缺陷
  - 分析故障和线上缺陷
- **执行效率**
  - 特性从提交到发布只需要短时间
  - 使用部署流水线自动串接起发布过程中的大部分活动
  - 发布审批流程短，时间少，电子化
  - 发布分支不必逐条人工配置流水线

发布（3）

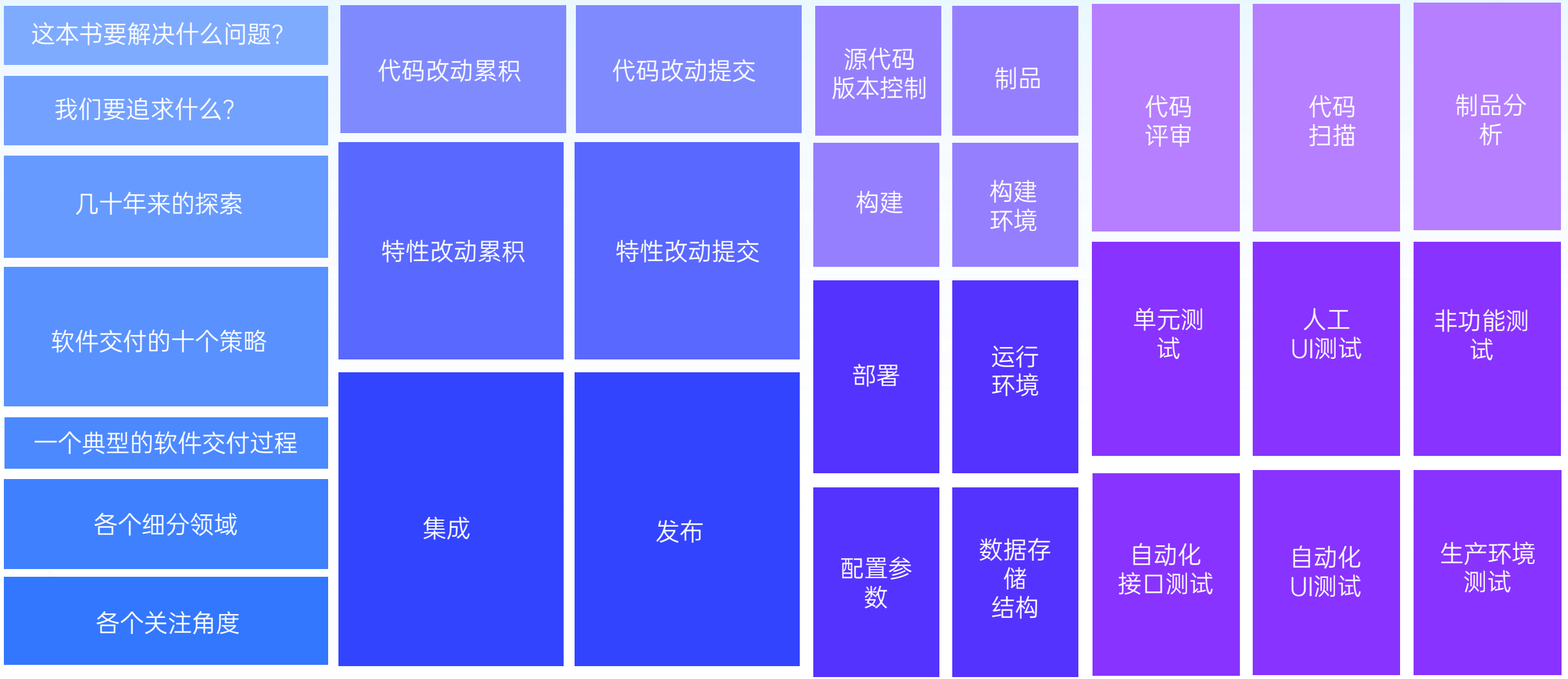
- **问题处理效率**
  - 故障恢复时长短
  - 线上缺陷修复时长短
  - 故障处理有明确适当的流程，通知合适的人处理
  - 发布回滚操作方便
  - 线上缺陷修复、紧急需求等情况，有紧急发布流程

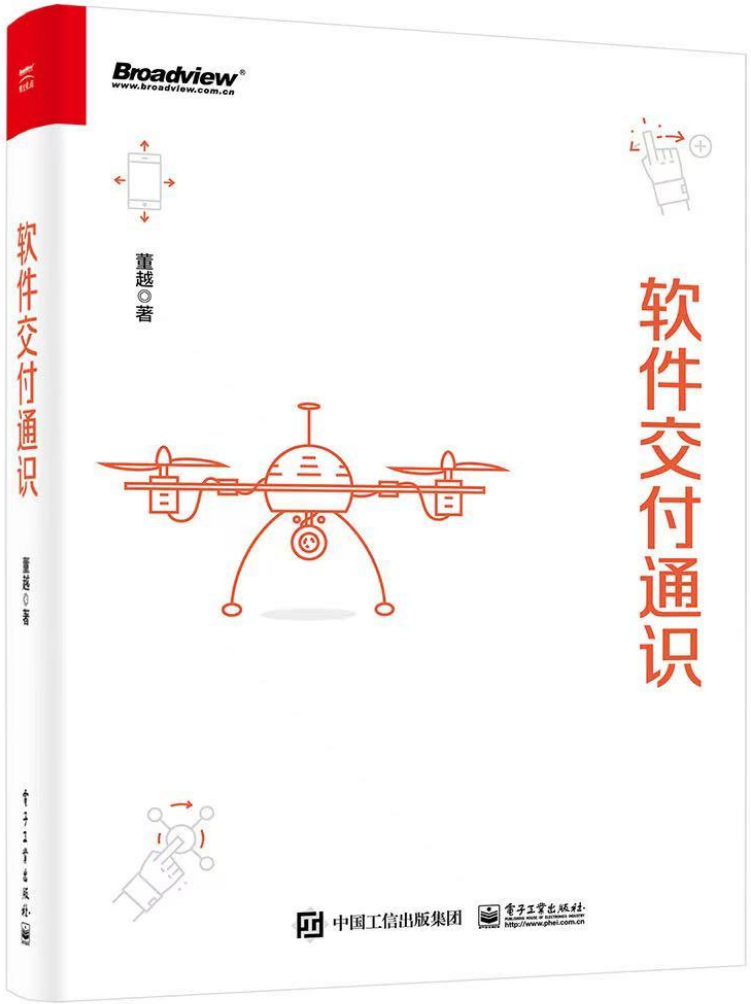
PART 4 了解各个具体活动


各具体活动
源代码及其构建
源代码版本控制
构建
构建环境管理
制品管理
部署运行
部署
运行环境管理
配置参数管理
数据存储结构管理
静态测试
代码评审
代码扫描
制品分析
动态测试
单元测试
自动化接口测试
人工UI测试
自动化UI测试
非功能测试
生产环境测试

PART 5 如何了解更多

《软件交付通识》


这本书要解决什么问题?
我们要追求什么?
几十年来的探索
软件交付的十个策略
一个典型的软件交付过程
各个细分领域
各个关注角度
代码改动累积
代码改动提交
特性改动累积
特性改动提交
集成
发布
源代码
版本控制
制品
构建
构建
环境
部署
运行
环境
配置参
数
数据存
储
结构
代码
评审
代码
扫描
制品分
析
单元测
试
人工
UI测试
非功能测
试
自动化
接口测试
自动化
UI测试
生产环境
测试



京东

THANKS