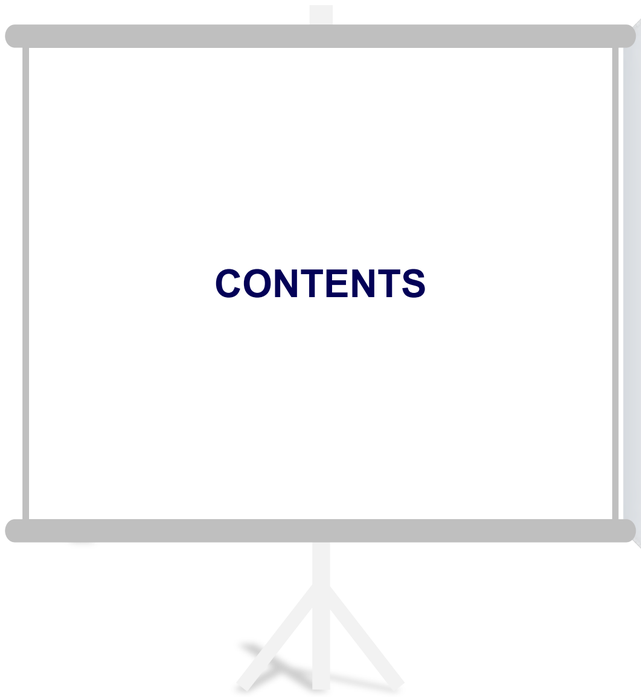
MTSC 2021 中国互联网测试开发大会 深圳站
TESTING SUMMIT CONFERENCE CHINA 2021

前端流量回放在企业应用的落地

ByteDance 字节跳动 高亚



CONTENTS

/01 背景介绍

为什么要做Echo前端流量回放？

痛点分析

对方正在输入中...

前端：又要写前端自动化case啦，要写好久呀！

测试：发版也太频繁了，每天都在回归！

测试：功能越来越多了，功能都回归不过来啦！

前端：冒烟case好多，自测时间不够啦！

- 前端变更频繁
- UI自动化编写维护成本高
- UI自动化稳定性差
- UI自动化对样式校验差
- 手工回归效率低

技术调研

Rrweb

- 基于DOM的变化的录制
- 可以重播整个web的变化
- 「录屏&录DOM」，用于问题复现
- 未记录后端请求
- 需要配套Rrweb-player回放
- 回放时静态资源受测试数据影响

Timecat

- 基于浏览器原生事件录制
- 持久化到浏览器内存空间IndexedDB
- 未记录后端请求
- 回放时静态资源受测试数据影响较大
- 主要用于问题复现
- 处于WIP阶段

Headless Recorder

- Chrome插件，录制操作行为
- 生成puppeteer脚本

其他类Chrome插件

- 可以记录Http请求
- 可以记录DOM变化
- 只支持Chrome浏览器

问题拆解

回放&录制环节拆解

如何搭建录制&回放的模型？

应该拆解为哪些模块？

前端行为轨迹抓取

上报前端事件和所有后端请求标识

后端接口抓取

跨应用调用，尽可能录制

精准回放

多种方式定位元素，多种策略保证接口Mock成功率

问题快速定位

Diff降噪，快速处理异常图片，减少分析成本

/02 总体设计

Echo架构设计、实现设计、模块介绍

总体设计-架构图


显示层
权限管理
配置管理
录制管理
回放管理
业务层
Echo管理端
RepeaterService
UiCaseCore
执行层
UiCaseRunner
EchoSDK-VUE
EchoSDK-React
RepeatAgent
存储层
MySQL
RocketMQ
Redis
ElasticSearch
文件存储服务
第三方服务
权限服务
消息中心
环境治理
图片Diff
图片聚合
PassPort

总体设计-重点模块介绍

录制

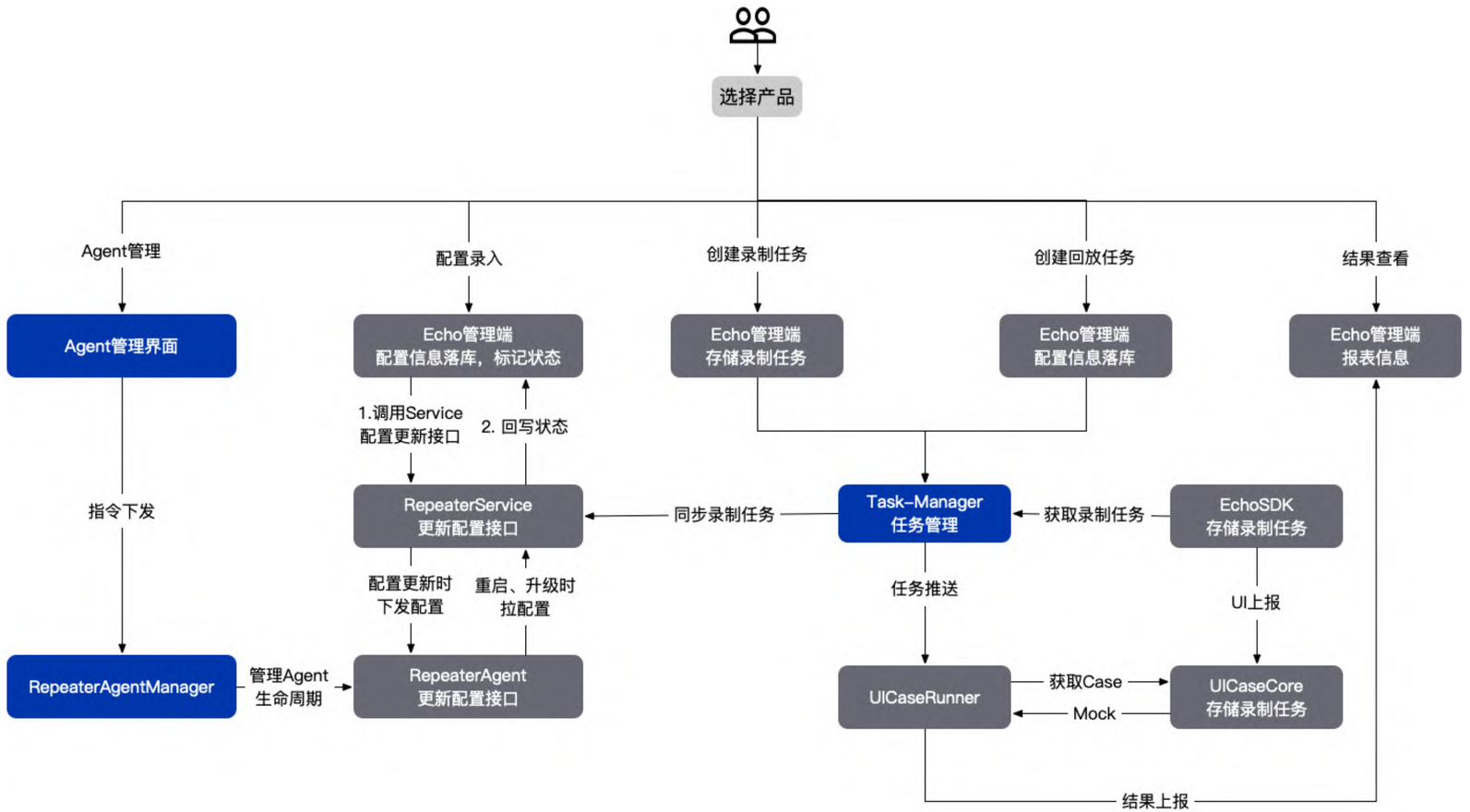
- EchoSDK: 前端埋点上报SDK
- RepeaterAgent: 后端接口录制Agent
- RepeaterService: 管理并下发RepeaterAgent配置

回放

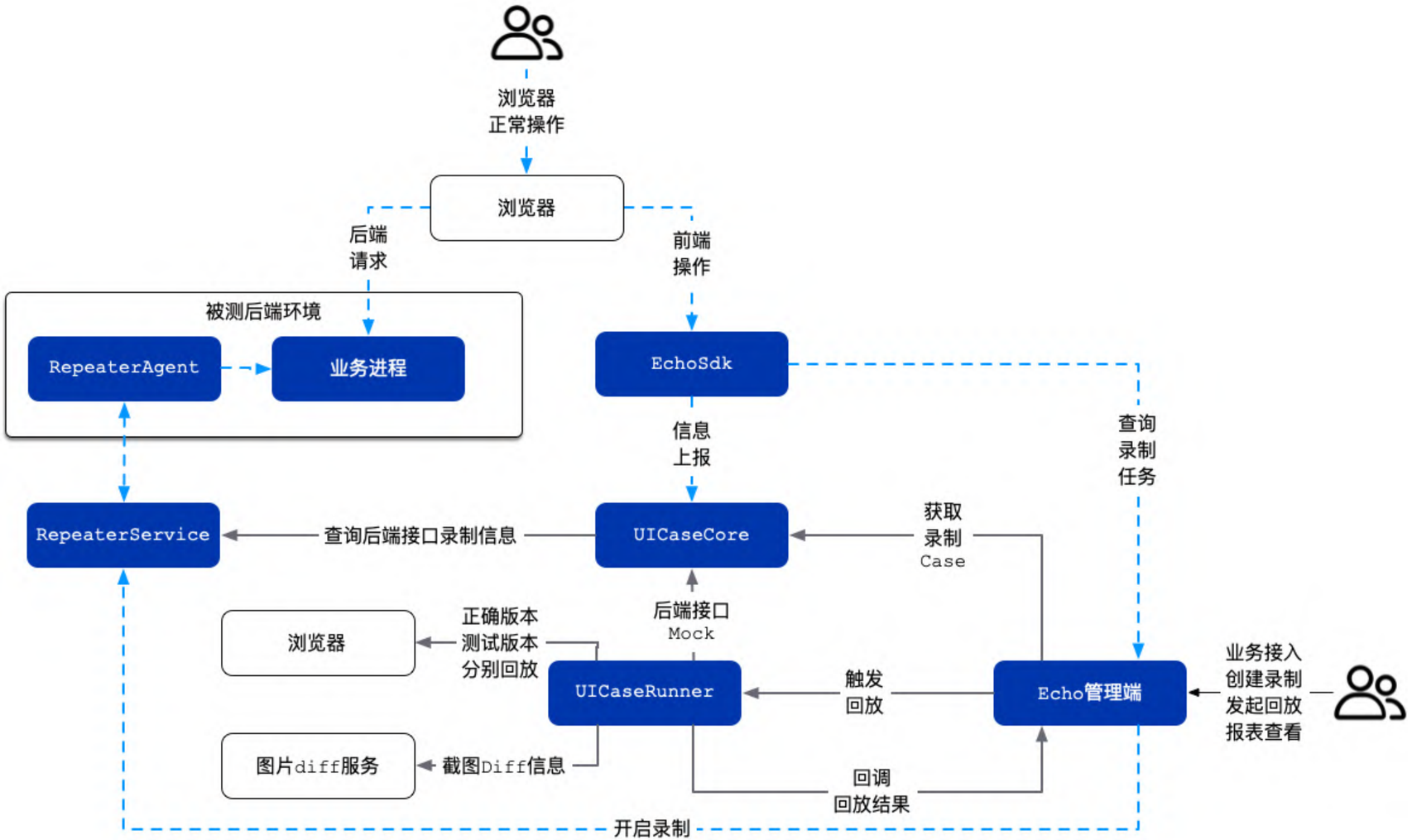
- UICaseCore: 接收EchoSDK上报信息，组装回放Case，后端接口Mock
- UICaseRunner：前端回放执行Agent

任务管理

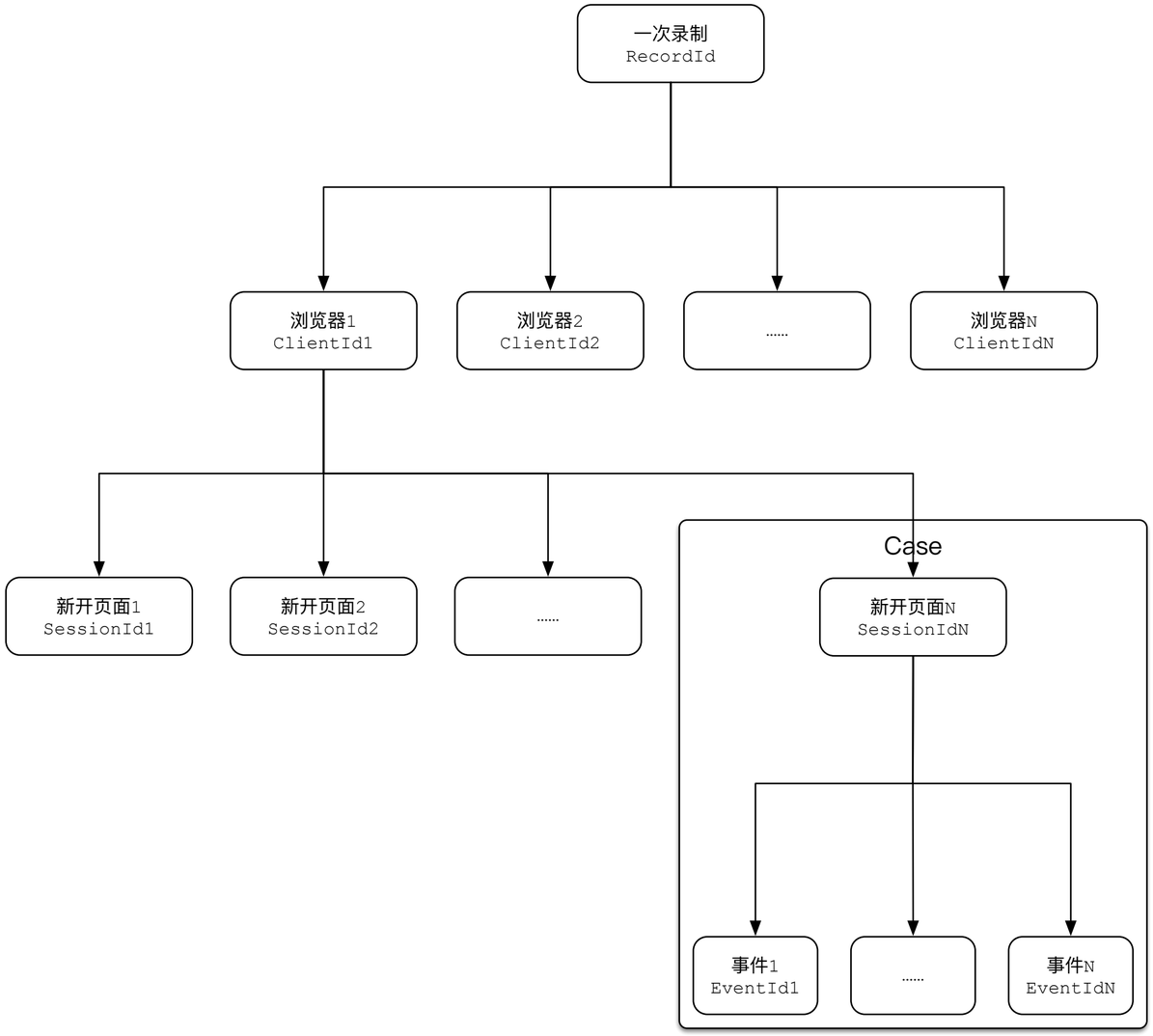
- Echo管理端: 录制、回放任务开启和报表展示
- Task-Manager: 任务管理模块
- Repeater Agent Manager：Agent生命周期管理



总体设计-系统交互


浏览器
正常操作
浏览器
后端
请求
前端
操作
被测后端环境
RepeaterAgent
业务进程
EchoSdk
信息
上报
查询
录制
任务
RepeaterService
查询后端接口录制信息
UICaseCore
获取
录制
Case
后端接口
Mock
浏览器
正确版本
测试版本
分别回放
UICaseRunner
触发
回放
Echo管理端
业务接入
创建录制
发起回放
报表查看
图片diff服务
截图Diff信息
回调
回放结果
开启录制

实现设计-EchoSDK



Request Payload view source


```
▼ {clientId: "b27d6756-71af-4d15-9e9a-b857c442b6f8", sessionTime: 1619775919362,…}
  clientId: "b27d6756-71af-4d15-9e9a-b857c442b6f8"
  ▼ element: {,…}
    selector: "html>body>div>div:nth-of-type(2)>div:nth-of-type(2)>div>div>div>div>div>div>div>div:nth-of-type(2)
  ▼ event: {options: {duration: 1351, x: 1122, y: 0}, type: "scroll"}
    ▼ options: {duration: 1351, x: 1122, y: 0}
      duration: 1351
      x: 1122
      y: 0
    type: "scroll"
  eventId: "1de6314d-2904-47fa-b9f1-7a6139a703bf"
  eventTime: 1619775920770
  location: "https://[redacted]"
  pointerPositions: []
  recordId: 432
  ▼ requests: ["0e12d650-4f2c-4bda-9234-1d236c307517", "9ac8b580-088d-41cf-9266-6be75994d8c2",…]
    0: "0e12d650-4f2c-4bda-9234-1d236c307517"
    1: "9ac8b580-088d-41cf-9266-6be75994d8c2"
    2: "ee496324-617e-4eac-a8e6-a9948700130a"
    3: "73331986-f42e-4641-aa2d-cde49dbd040a"
    4: "1adbe71f-8f3e-4618-acca-00a255fd5fce"
    5: "b99bf91b-5272-4d33-955c-6f959c76dd70"
    6: "820c6841-6d53-4cc4-af4d-ef5ece7516fe"
  sessionId: "f943bf49-d033-4294-b79f-50608a5e8c1e"
  sessionTime: 1619775919362
```

实现设计-EchoSDK

window.addEventListener

Hook用户事件

输入

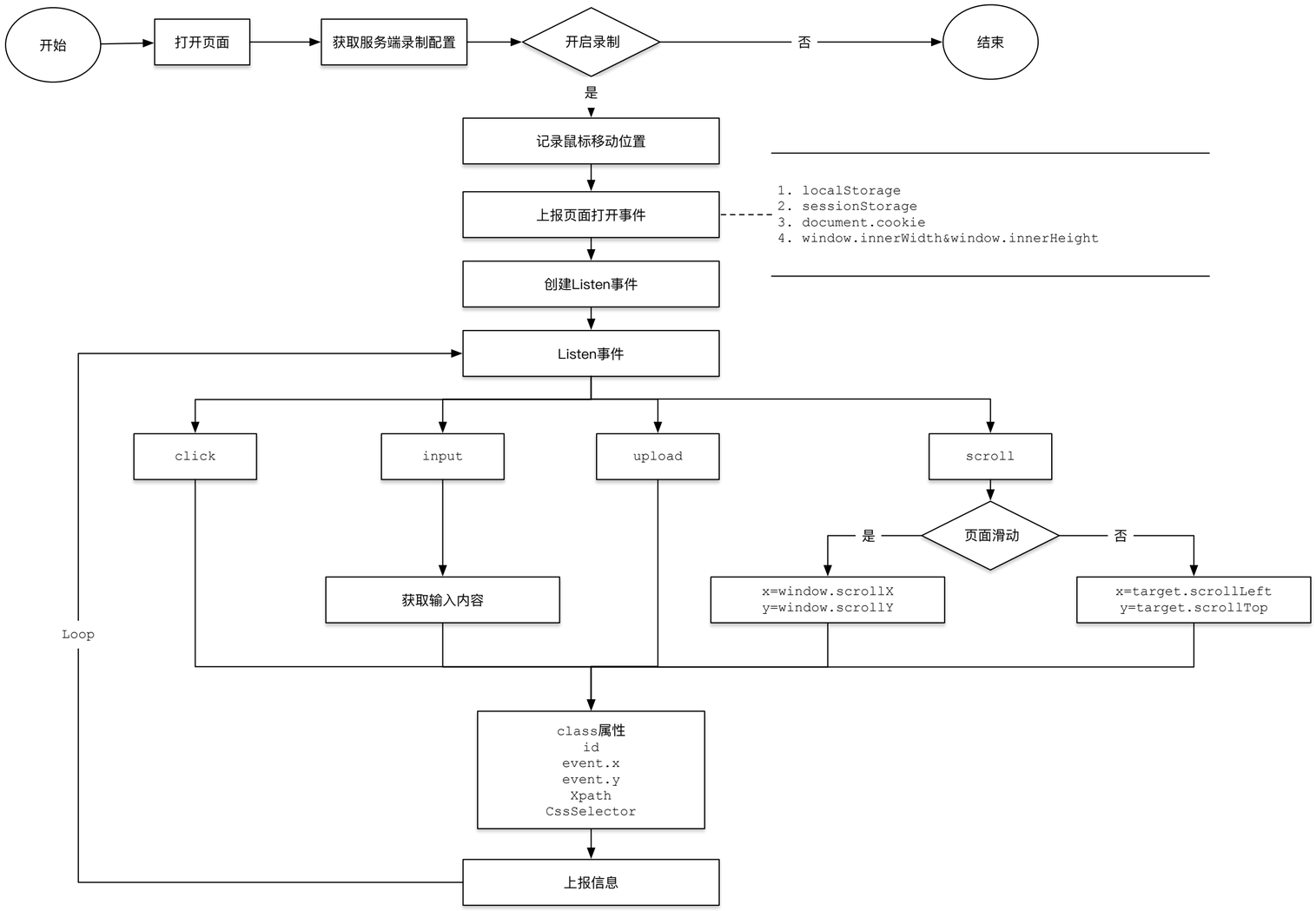
- compositionstart
- compositionend
- keydown

点击

滑动

文件上传

- event.target.tagName === ‘INPUT’ && event.target.type = == ‘file’
- Listen change事件，先检测 target 是否为 input[type=file]



实现设计-EchoSDK

组件加载顺序

组件加载顺序错误，会导致TraceId 添加失败，从而导致Mock失败，影响回放

- XMLHttpRequest.prototype.open
- XMLHttpRequest.prototype.send
- window.fetch
- 增加TraceId

```
const origOpen = XMLHttpRequest.prototype.open
XMLHttpRequest.prototype.open = function (method, url) {
    // 过滤无需录制的接口
    // url增加唯一请求标识
    origOpen.call(this, method, newUrl)
};
```

```
const originSend = XMLHttpRequest.prototype.send
MLHttpRequest.prototype.send = function (body) {
    // 过滤无需录制的接口
    // 此时Url里面无需增加唯一标识，因为Open里面已经增加过了
    originSend.call(this, body)
};
```

浏览器初始化信息上报

- 窗口大小
- 登录态
- 侧边栏收缩状态
- 业务配置信息

```
const originFetch = window.fetch
window.fetch = function (input, options = {}) {
    // 缓存请求信息
    // 如果input是Object，请求地址就是 input.url，否则input就直接是请求地址
    // 过滤一些无需hook的请求
    // 如果input不是Object，是地址，则增加请求唯一标识后，直接调用originFetch
    // 如果input是Object，需要手工改写input，再调用originFetch，改写内容如下:
const newInput = new Request(shouldUseHeader ? realUrl : urlWithQuery, {
                        method: input.method,
                        headers: input.headers,
                        body: body,
                        referrer: input.referrer,
                        referrerPolicy: input.referrerPolicy,
                        mode: input.mode,
                        credentials: input.credentials,
                        cache: input.cache,
                        redirect: input.redirect,
                        integrity: input.integrity,
                    })
```

组件规范

- 文件上传

实现设计-后端接口录制

01

Java后端接口抓取

- RepeaterService： JVM-Sandbox
- RepeaterAgent： JVM-Sandbox-Repeater

02

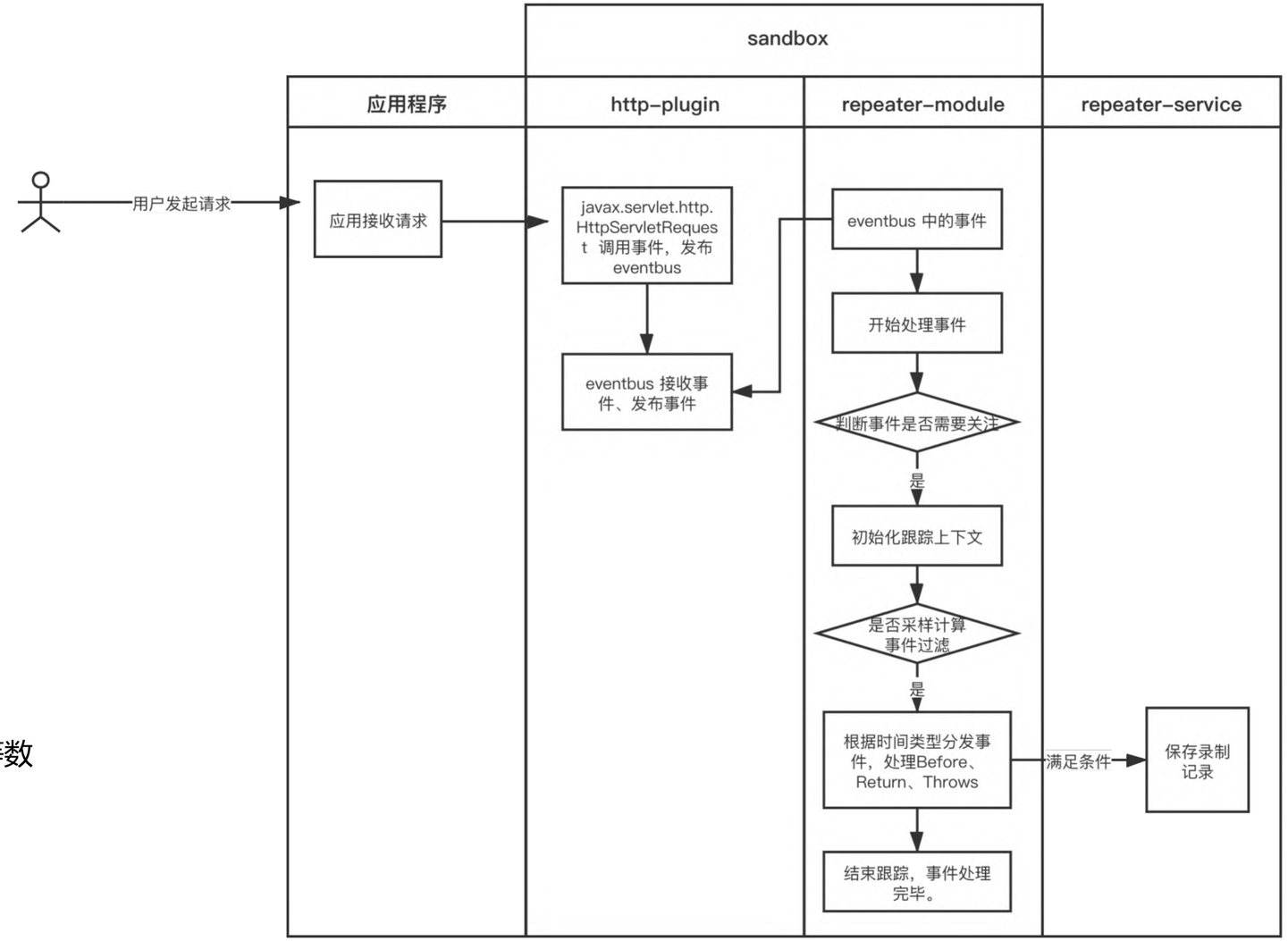
二次开发的功能

- 开发sidecar agent，减少接入成本
- 采集服务关键信息
- 支持跨环境录制
- 重构http插件
- 存储迁移到ES

03

注意事项

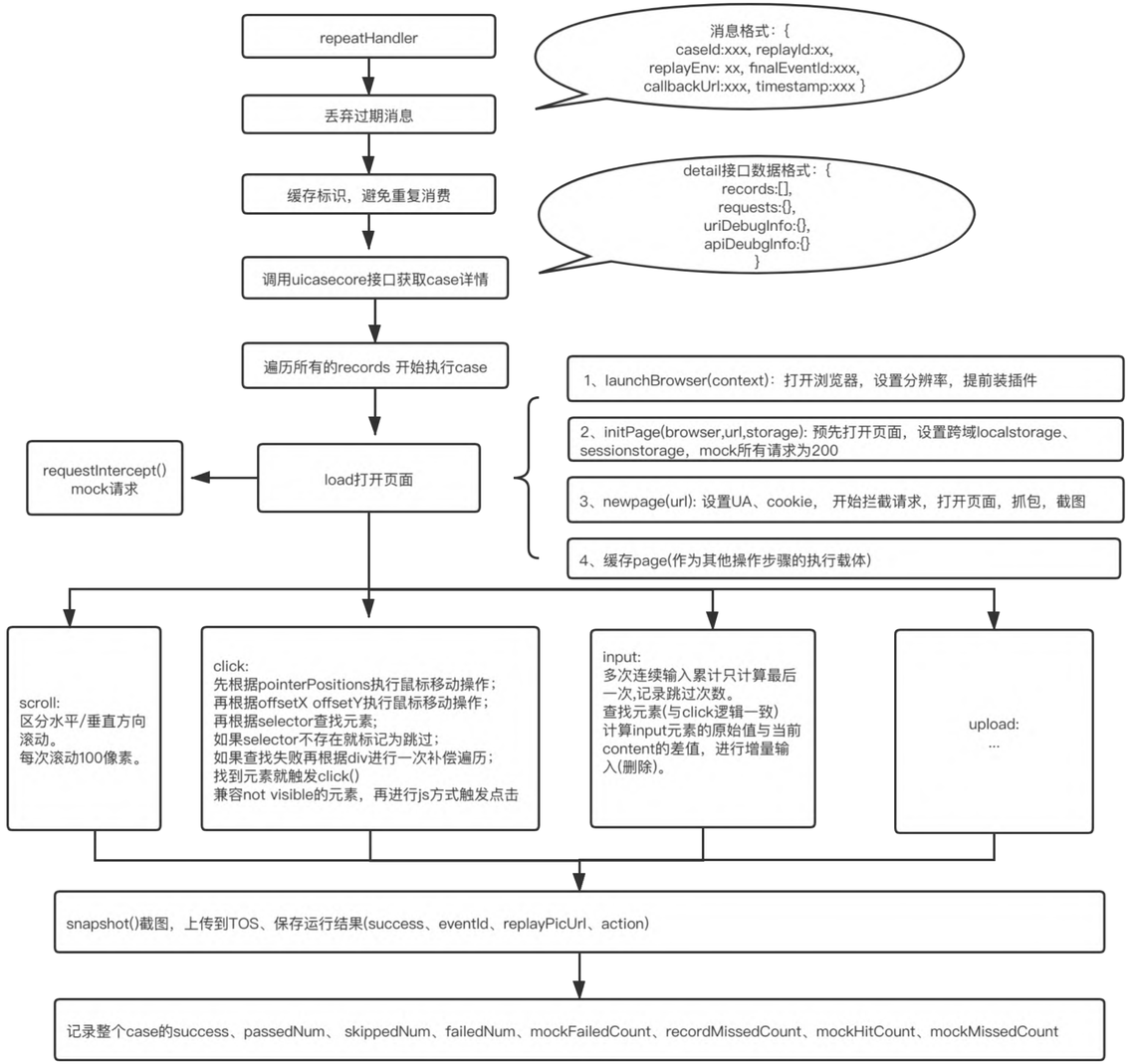
- 业务线的自建网关服务可能会影响requestURI等数据采集
- 录制期间对业务系统带来的性能开销



实现设计-UiCaseRunner

注意点

- 在回放前提前开一个页面，设置LocalStorage和SessionStorage
- 鼠标hover
- Input注意中文输入，只计算最终态
- 动态元素
- 回溯查找
- 非可见区元素点击：js代码实现



实现设计-UiCaseRunner

01

浏览器

- Cookie
- Local Storage
- Window Size
- 请求拦截

行为回放

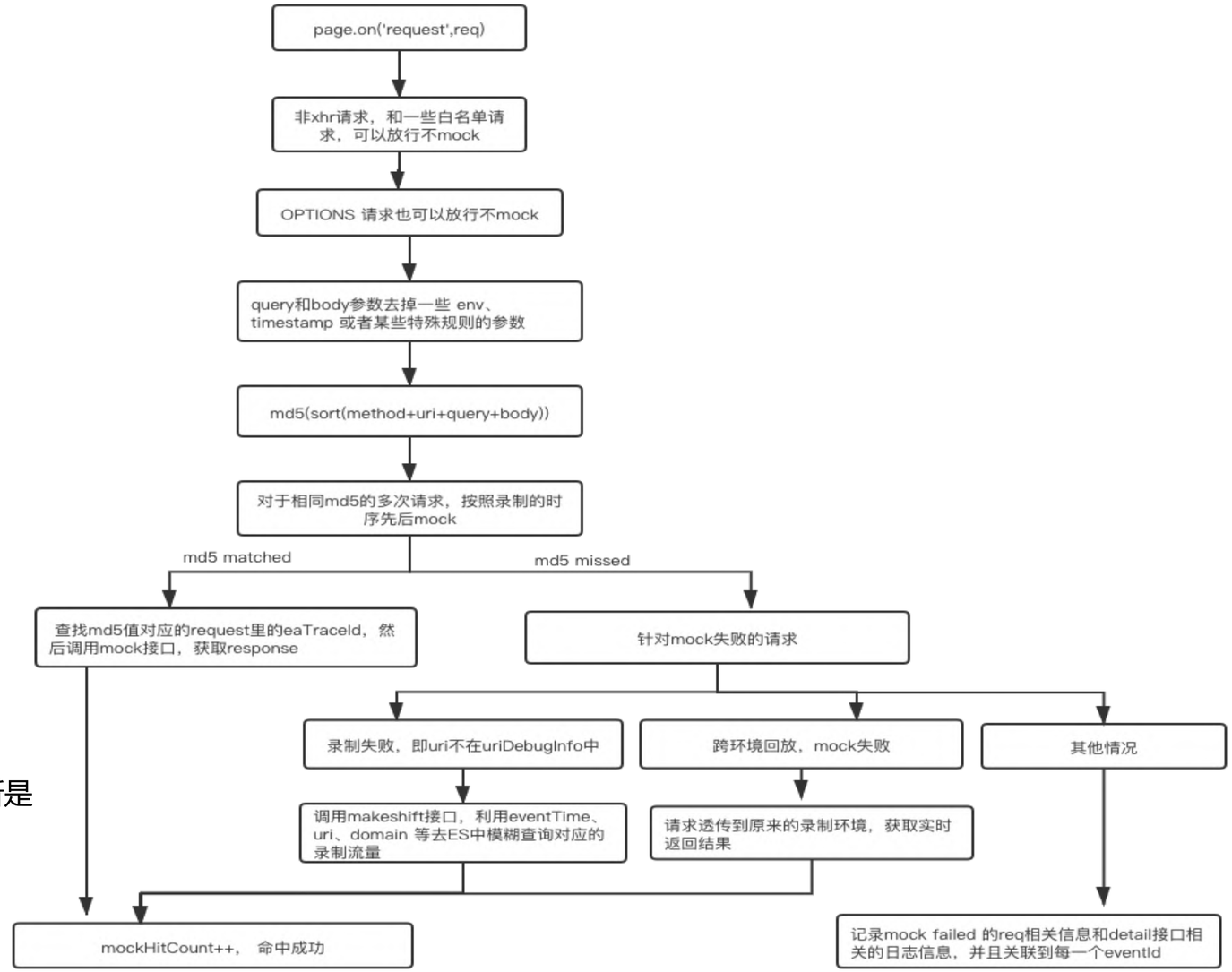
02

- 输入
- 点击
- 滑动
- 文件上传
- 鼠标move/hover

Mock 策略

03

- 根据 md5(sort(method+uri+body+query)) 判断是否mock成功
- Mock失败的补偿机制：模糊查询，请求透传



实现设计-UiCaseCore

01

基础信息

- Cookie
- Local Storage
- Window Size
- Case信息

02

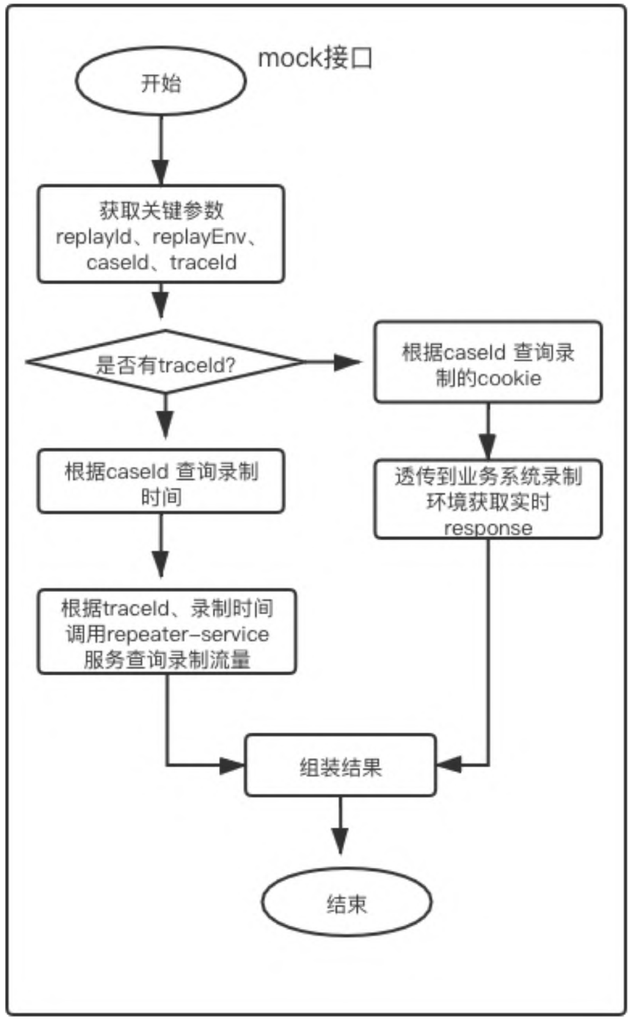
接口Mock

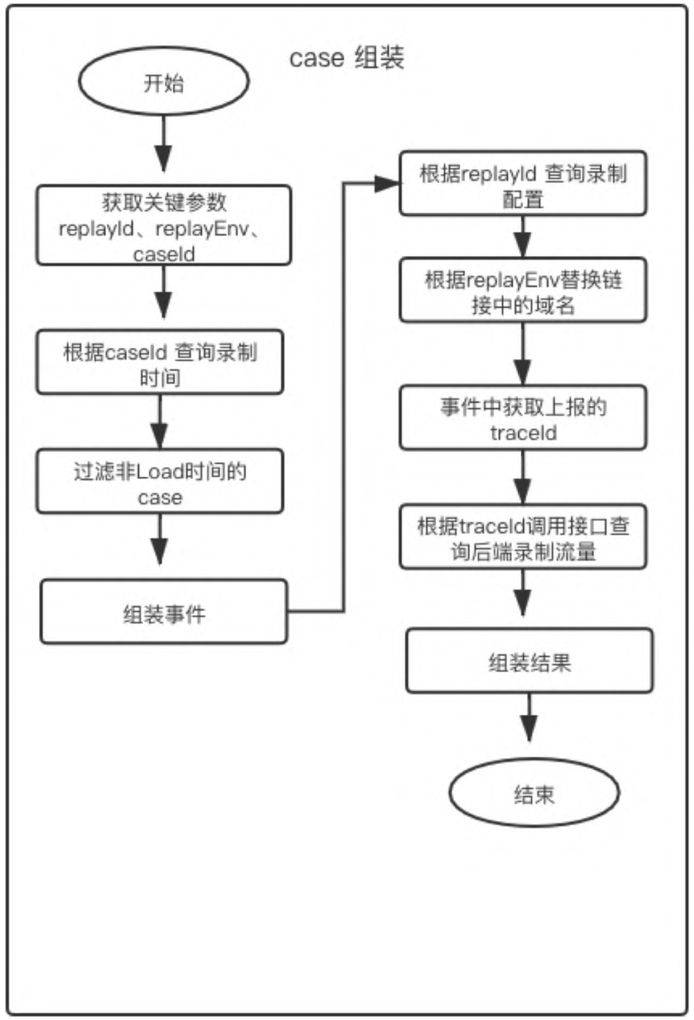
- 唯一标识
- 接口请求顺序
- UrlEncode

03

Case组装

- 前端上报
- 根据页面筛选
- 过滤异常上报事件
- 域名替换





实现设计-管理端报表内容

任务执行状态定义

任务执行失败

【平台操作失败】

- Mock失败
- 录制失败
- Diff失败
- 基准执行失败
- 测试执行失败

任务执行成功

- 【Diff通过】
  任务执行通过，且图片Diff值高于设定阈值
- 【Diff比对未通过】
  任务执行通过，但图片Diff值低于设定于阈值
- Diff聚合

实现设计-管理端报表内容

任务执行效果评估

回放执行成功率

回放执行成功步骤数/回放步骤总数

成功步骤数量（包括系统跳过步骤）

失败步骤数量

总步骤数量

Mock成功率

基准环境Mock成功率

基准环境Mock成功/Mock总数

测试环境Mock成功率：

测试环境Mock成功/Mock总数

图片聚合效果

Diff成功率

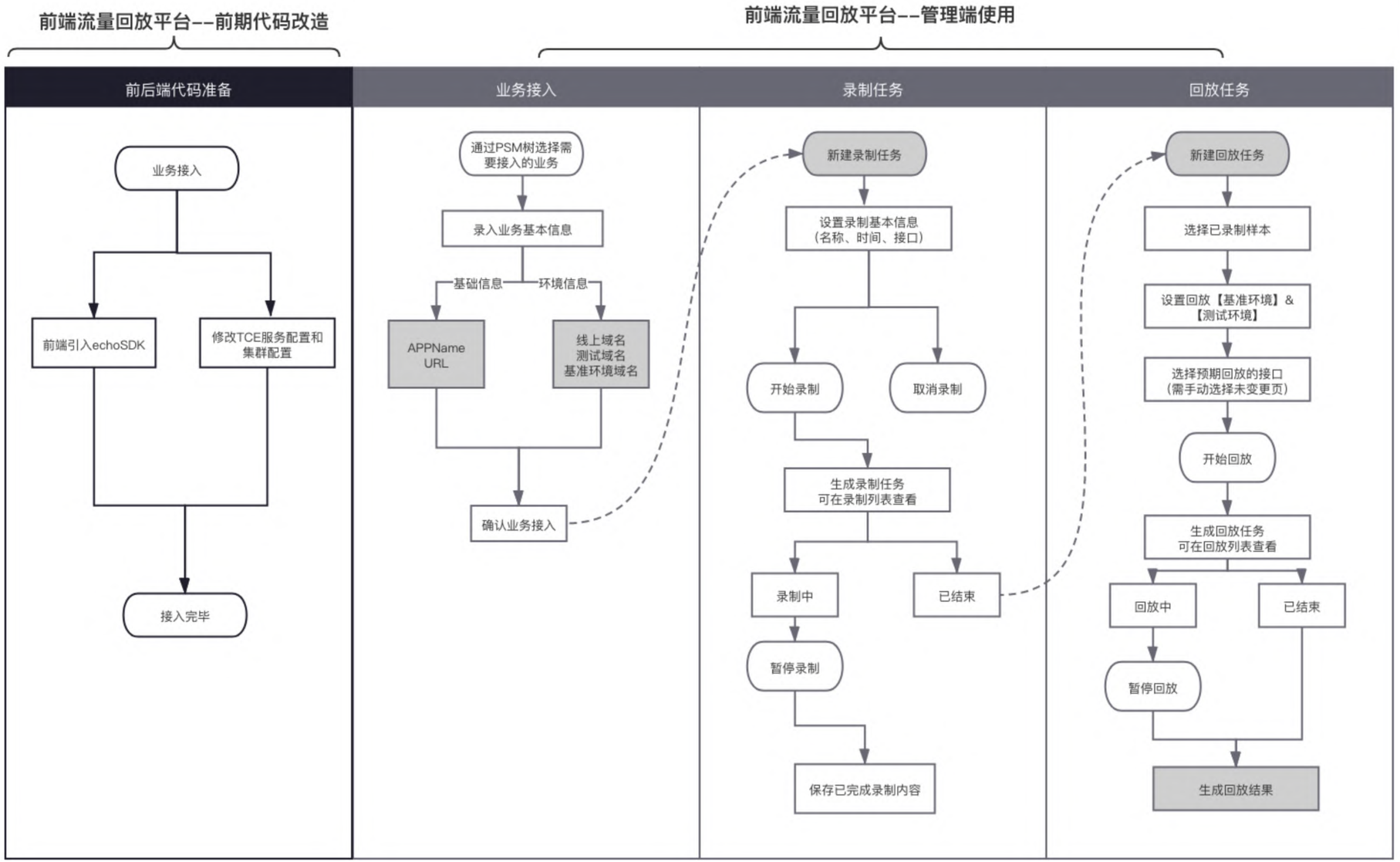
Diff成功数量/Diff图片总数

Diff聚合数量

/03 进展效果

Echo业务进展&落地效果

进展效果-用户操作流程


前端流量回放平台--前期代码改造
前端流量回放平台--管理端使用
前后端代码准备
业务接入
录制任务
回放任务
业务接入
前端引入echoSDK
修改TCE服务配置和集群配置
接入完毕
通过PSM树选择需要接入的业务
录入业务基本信息
基础信息
环境信息
APPName URL
线上域名 测试域名 基准环境域名
确认业务接入
新建录制任务
设置录制基本信息（名称、时间、接口）
开始录制
取消录制
生成录制任务 可在录制列表查看
录制中
已结束
暂停录制
保存已完成录制内容
新建回放任务
选择已录制样本
设置回放【基准环境】&【测试环境】
选择预期回放的接口（需手动选择未变更页）
开始回放
生成回放任务 可在回放列表查看
回放中
已结束
暂停回放
生成回放结果

进展效果-回放结果页

回放结果 任务接入信息

确认bug			
待处理bug	非bug	重复bug	已聚合bug
8	3	0	95(已完成)

回放执行成功率		
失败步骤数	成功步骤数	总步骤数
195	1579	1774

89.21%

平均比对通过率	
有效步骤总数	成功步骤数
674	546

81.01%

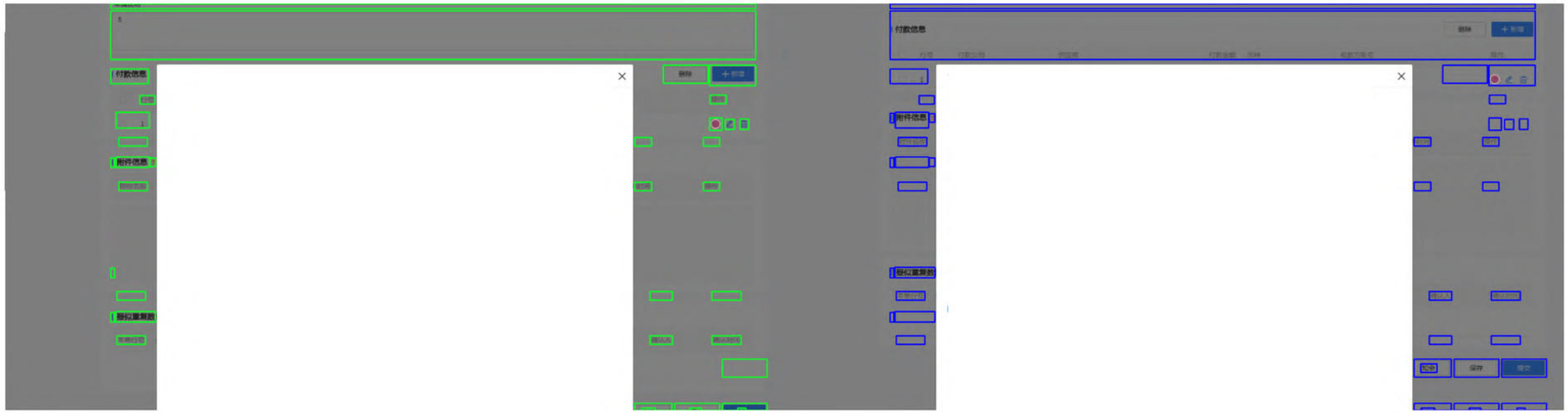
接口mock成功率	
基准环境成功率	测试环境成功率
94.15% (934/992)	95.10% (950/999)

Case结果详情

有diff的图片　全部case

待处理(8)　确认bug(2)　非bug(3)　重复bug(0)

回放基准环境: pre　回放测试环境: boe



步骤标识: 27d283f9-5abc-478a-9df5-7266873...

比对相似度: 91%

待处理

非bug(业务变更)

非bug(Echo异常)

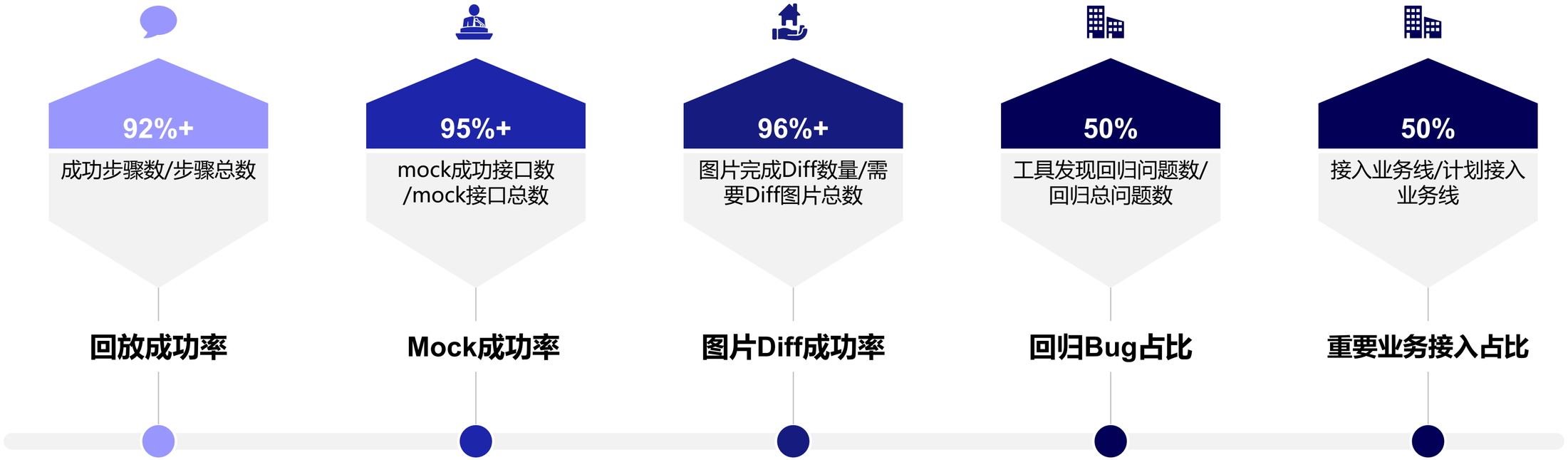
重复bug

查看全部步骤 >

Case Id: 76697

总步骤: 5,成功步骤: 5,有diff步骤: 5

进展效果


92%+
成功步骤数/步骤总数
回放成功率
95%+
mock成功接口数/mock接口总数
Mock成功率
96%+
图片完成Diff数量/需要Diff图片总数
图片Diff成功率
50%
工具发现回归问题数/回归总问题数
回归Bug占比
50%
接入业务线/计划接入业务线
重要业务接入占比

进展效果-关键指标

回放成功率.

92%+ + 84% ↑

回放成功率由50%提升到92%

Mock成功率.

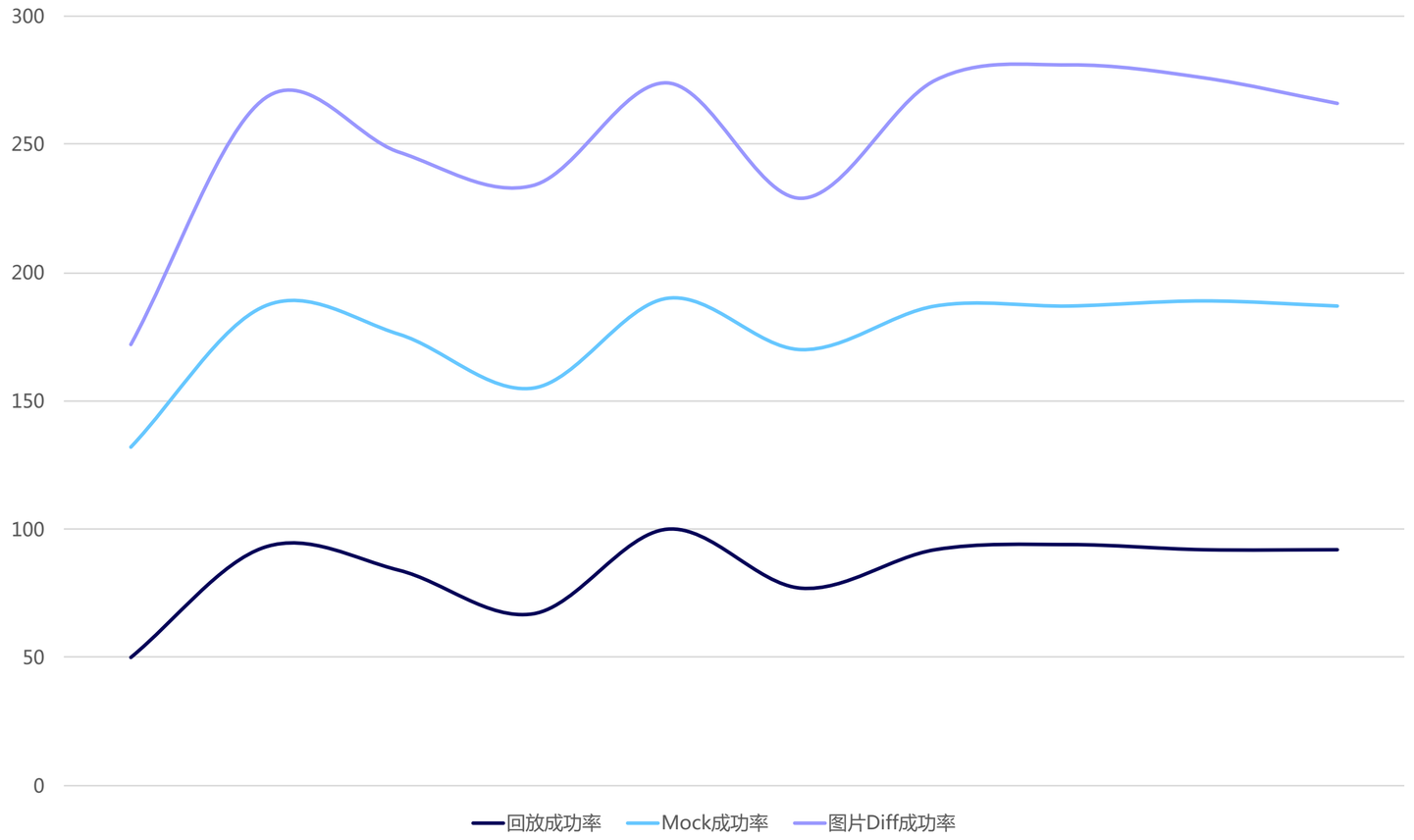
95%+ + 19% ↑

Mock成功率由80%提升到95%

图片Diff成功率.

96%+ + 58% ↑

图片Diff成功率由60%提升到96%



进展效果-解决Chrome有头无头模式差异，提升图片Diff成功率

01

A.

追查过程

- 排查Chrome、Node、Puppeteer版本
- 截图时机：执行操作之后Sleep 5s仍然出现
- 字体问题：设置成相同的字体
- Chrome参数配置：
- --font-render-hinting=none|medium
- --enable-font-antialiasing
- 调大图片Diff参数： pixelmatch threshold
- Chrome无头和有头模式下的差异

B.

解决效果

- 相同的Case回放：A产品线的图片diff通过率由62%提升到88%
- 相同的Case回放： B产品线的图片diff通过率由82%提升到90%

If you want Chrome Headless to closely emulate Chrome, here's the code to fix it:

```
1  const browser = await puppeteer.launch();
2  const page = await browser.newPage();
3
4  const headlessUserAgent = await page.evaluate(() => navigator.userAgent);
5  const chromeUserAgent = headlessUserAgent.replace('HeadlessChrome', 'Chrome');
6  await page.setUserAgent(chromeUserAgent);
7  await page.setExtraHTTPHeaders({
8    'accept-language': 'en-US,en;q=0.8'
9  });
```

puppeteer-headless.js hosted with ❤ by GitHub view raw

进展效果-自主拉起前端环境，解决前端发版不稳定导致回放成功率低的问题

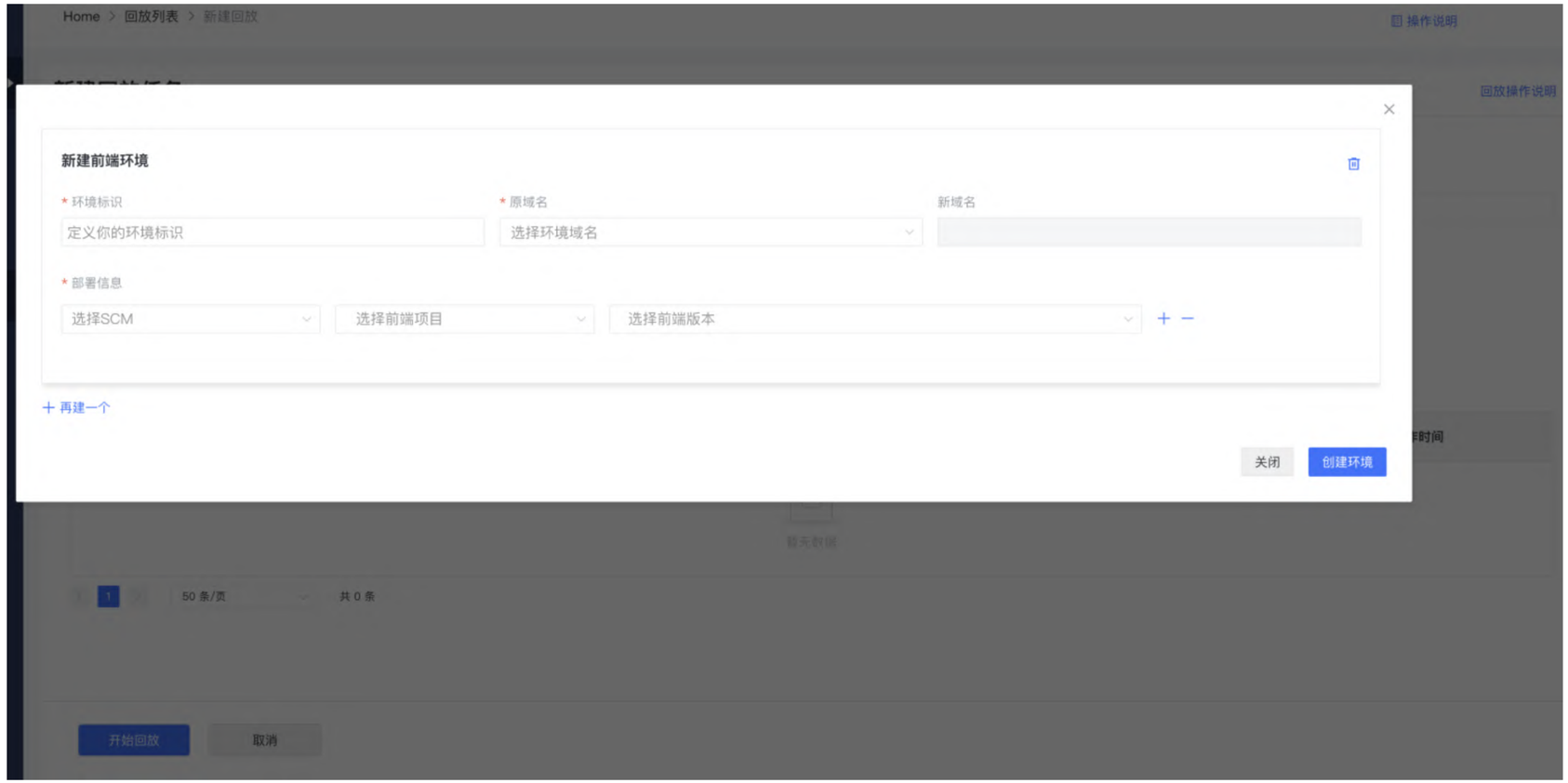
02

问题：

- 研发发版不稳定导致回放成功率较低
- 一个产品，存在多个微前端服务

解决方案：

- 明确展示基准和测试环境的版本信息
- 自动拉起新的前端测试环境
- 复用已拉起的测试环境



进展效果-聚合相近的疑似bug图片，提升问题定位效率

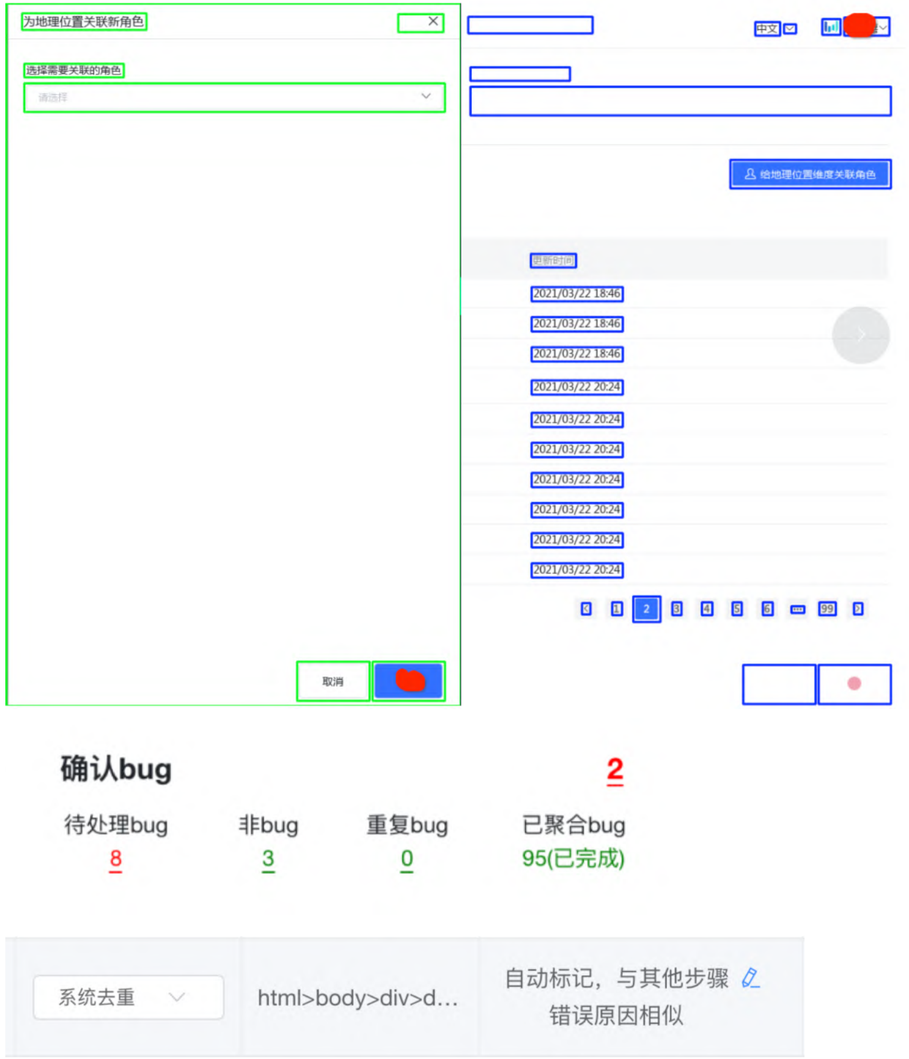
03

问题：

- 步骤数较多时，回放时间长
- 回放结果中存在多张相同的疑似bug图片，用户确认图片工作量较大

解决方案：

- 不同的Case，并行执行
- 同一个Case，基准和测试环境串行执行
- 截图中体现鼠标操作的轨迹
- 相同的步骤，在基准和测试环境截图
- 相似度小于95%则不通过
- 标注出Diff差异处
- 聚合Diff不通过的图片，根据相似度聚类
- 提供mock失败的日志查询



/04 未来规划

Echo会在哪些方向深耕，在哪些方向还有探索机会？

未来规划

多场景支持

- Hook更多浏览器事件
- 浏览器兼容性
- 页面性能
- 持续集成

加强与后端联动

- 前后端同时回放
- 后端拓扑图，提高Mock成功率
- 拉起干净的后端环境

Case智能筛选

- Case降噪
- Case精准回放
- Case 语义化
- Case debug

快速定位问题

- 实时回放
- 回放日志精细化
- AI识别缺陷分类
- 前端覆盖率

MTSC 2021 中国互联网测试开发大会 深圳站

TESTING SUMMIT CONFERENCE CHINA 2021

ByteDance 字节跳动

主办方：TesterHome


谢谢
THANKS