



# 商业银行UI自动化测试 技术演进及实践分享

东软集团

殷坤 · 2023年6月2日



# 目录

## CONTENTS

1. 商业银行业UI自动化测试现状
2. 传统自动化测试常见模式及瓶颈
3. AI技术在自动化测试领域的应用
4. 自动化测试推广实施实践



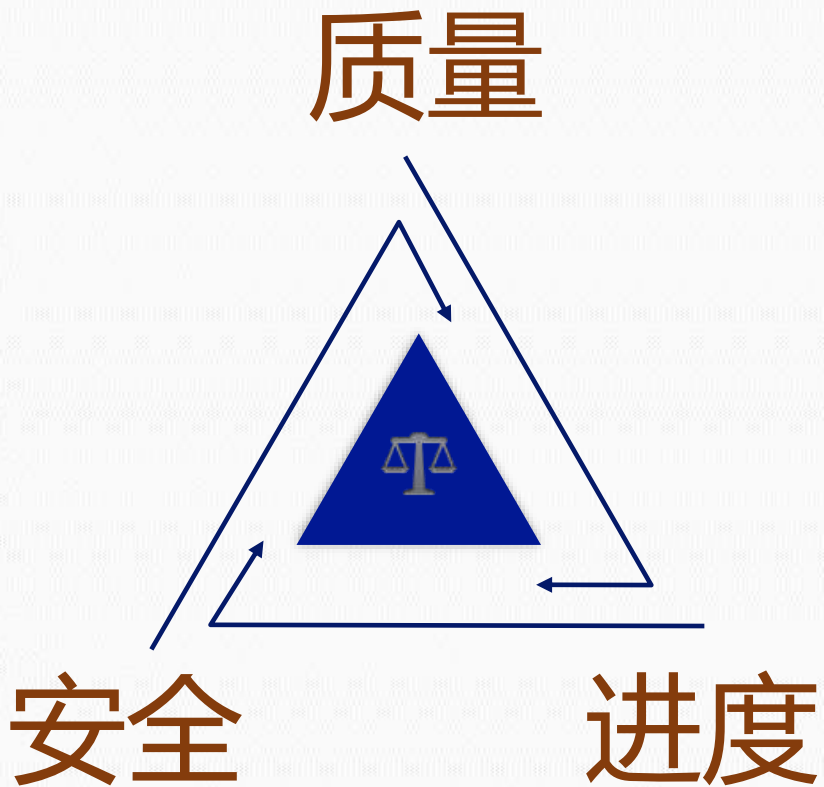


# 商业银行业UI自动化测试现状



# 银行业系统测试特点

❏ 银行作为金融行业的IT先进生产力代表，比传统行业重视安全、比互联网行业重视质量，同时又要保证进度。



测试工作量多、人员规模大



测试环境复杂、接入渠道多



被测系统多、技术架构复杂



重视DevOps中的CI环节



不太适用灰度发布、AB测试







# 银行业自动化测试现状



## CS桌面程序

版本迭代慢，环境单一，  
也一直缺乏可用工具，  
基本未曾开展自动化测试

QTP

## Web系统

出现主流的自动化工具，但各种  
UI框架的广泛应用加大了自动化  
测试的难度，一度处于暂停状态

Selenium

## 移动App

版本发布频繁，移动终端类  
型繁多，回归工作量加剧，  
是自动化测试重点投入方向

Appium

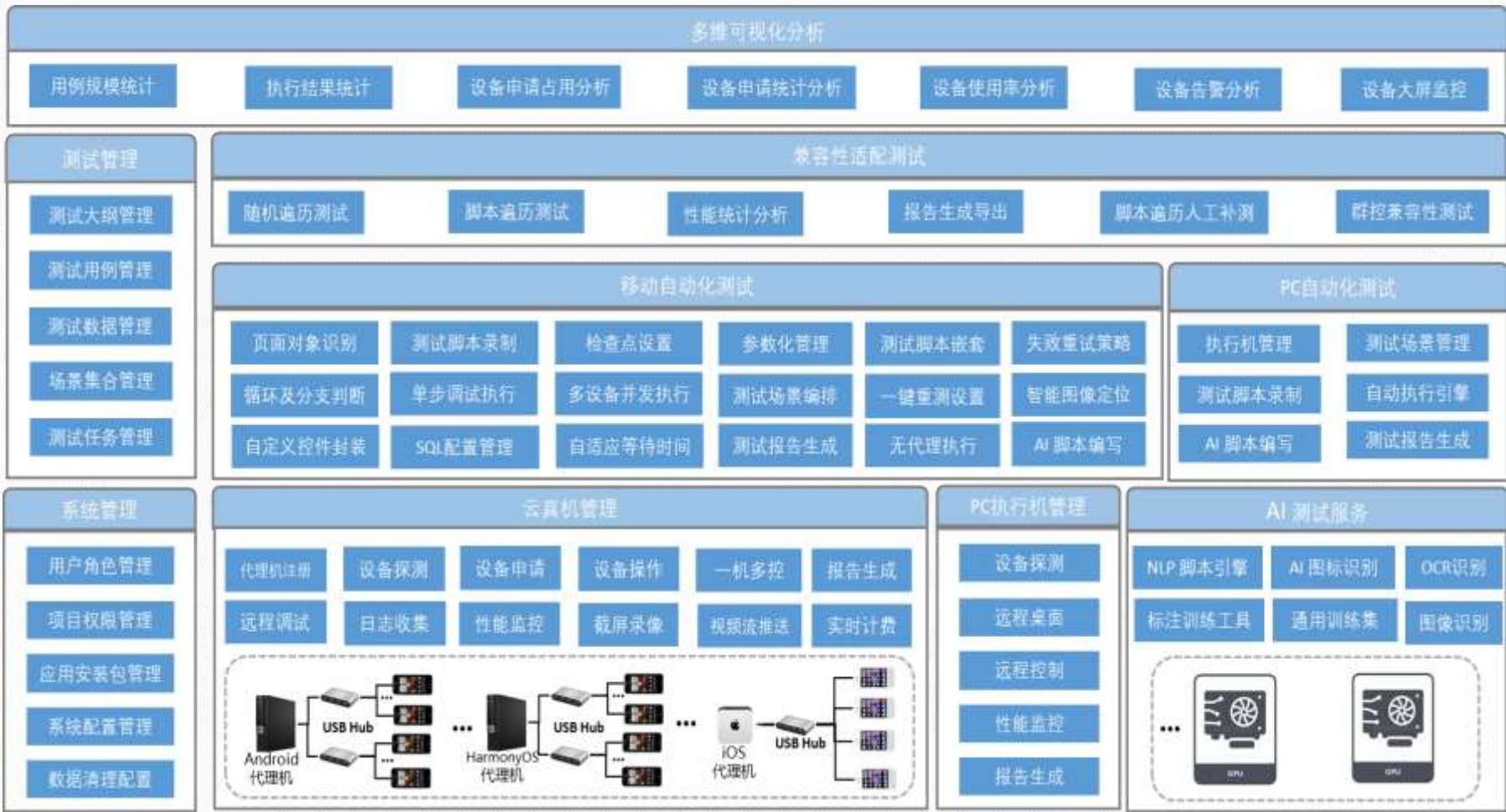
## 云原生应用

部署架构更复杂、版本发布  
更频繁，需要模拟各种环境  
故障，更需要自动化测试

深度学习、计算机视觉、OCR、自然语言等AI技术



# 银行业自动化测试平台发展趋势



01

## 多端统一化

移动端、Web端、桌面端采用统一的自动化测试平台。

02

## 引擎智能化

测试引擎采用深度学习技术实现智能化的对象识别和脚本解析。

03

## 手自一体化

不在单一的强调自动化测试，同时提供高效的手工功能测试和手工兼容性测试能力。

04

## 部署私有化

为了安全合规，测试环境进一步限制外部访问权限，自动化测试平台基本私有化部署。







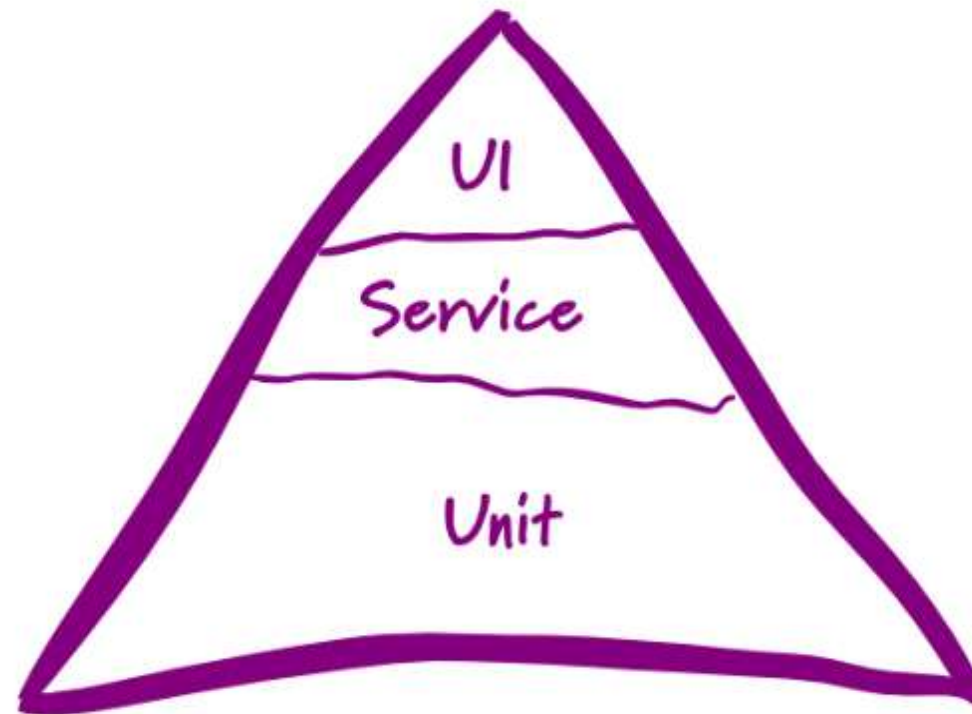
# UI端自动化测试的必要性及难点

□ UI测试是系统测试、用户验收测试的主要工作，具备如下特点：

- UI测试参与人员多、工作量大；
- UI测试相对比较机械重复、价值密度低；
- UI是最终用户唯一操作入口，直接影响用户体验和留存；
- UI端（客户端）环境差异会导致兼容性缺陷；
- Unit/Service层测试通常是缺乏的、不可信的；

□ UI端应该做自动化回归测试，但难点在于：

- UI自动化脚本稳定性较差，维护成本高；
- 测试团队确少编码技能，推广难度大；



分层自动化测试





# 传统自动化测试常见模式及瓶颈



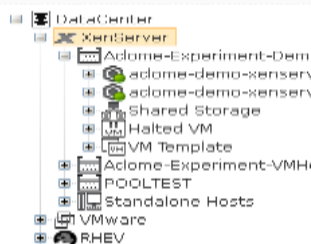


# 基于对象定位技术的UI自动化测试方案

## 1、成熟前端UI类库的广泛采用大大降低了开发成本和难度

代码值	代码标题
1	宁波
2	宁海

ddd  只能输入数字



	员工编号	姓名	职位	入职日期
1	1104	李红	技术总监	2005-08-08
2	1114	李洪	技术总监	2005-05-05
3	1105	赵薇	行政主管	2004-05-10
4	1105	张三妹	工程师	2007-11-08
5	1111	6555444	工程师	1991-08-04
6	1115	任我行	工程师	2003-03-18
7	1110	test	test	2007-10-01
8	1005	主表		

## 2、自动化测试脚本则要面对UI类库生成的海量源码

```
<div id="uniesp_form_TextBox_0" class="u-form-widget" widgetid="uniesp_form_TextBox_0">
  <div class="u-form-required" dojoattachpoint="requiredNode" style="visibility: visible;">*</div>
  <div class="u-form-field" dojoattachpoint="fieldNode">
    <div class="u-form-modified" dojoattachpoint="modifiedNode"></div>
    <div class="u-form-error" dojoattachpoint="errorNode" style="display: none;"></div>
    <div class="u-form-textbox-field">
      <input class="u-form-textbox" type="text" onfocus="uniesp_form_TextBox_0.focus()" dojoattachpoint="inputNode" focusNode="uniesp_form_TextBox_0" style="text-align: right; max-length: 20;">
    </div>
  </div>
</div>
```

```
<div id="uniesp_form_TextBox_0" class="u-form-widget" widgetid="uniesp_form_TextBox_0">
  <div class="u-form-required" dojoattachpoint="requiredNode" style="visibility: visible;">*</div>
  <div class="u-form-field" dojoattachpoint="fieldNode">
    <div class="u-form-modified" dojoattachpoint="modifiedNode"></div>
    <div class="u-form-error" dojoattachpoint="errorNode" style="display: none;"></div>
    <div class="u-form-textbox-field">
      <input class="u-form-textbox" type="text" onfocus="uniesp_form_TextBox_0.focus()" dojoattachpoint="inputNode" focusNode="uniesp_form_TextBox_0" style="text-align: right; max-length: 20;">
    </div>
  </div>
</div>
```

```
<div id="uniesp_form_TextBox_0" class="u-form-widget" widgetid="uniesp_form_TextBox_0">
  <div class="u-form-required" dojoattachpoint="requiredNode" style="visibility: visible;">*</div>
  <div class="u-form-field" dojoattachpoint="fieldNode">
    <div class="u-form-modified" dojoattachpoint="modifiedNode"></div>
    <div class="u-form-error" dojoattachpoint="errorNode" style="display: none;"></div>
    <div class="u-form-textbox-field">
      <input class="u-form-textbox" type="text" onfocus="uniesp_form_TextBox_0.focus()" dojoattachpoint="inputNode" focusNode="uniesp_form_TextBox_0" style="text-align: right; max-length: 20;">
    </div>
  </div>
</div>
```

## 3、测试脚本编写维护难度及工作量加大、自动化回放执行的稳定性及效率大幅降低

- 页面DOM结构非常复杂——所录制/编写脚本的复杂度变的更大、可读性变得更差；
- 即使页面代码没有任何变化，UI框架的升级也会导致DOM结构的变化——脚本无效的风险变得更大；
- 控件ID是自动生成的，甚至可能随机变化——导致根据ID定位控件的策略无效；







# 常见UI自动化测试设计模式的局限性

被测页面示例

自动化脚本示例



录制工具生成的脚本



基于开源类库编写的脚本

```
<method id="创建用户">
  <event id="[menu] 组织机构. 用户管理"/>
  <event id="[button] 新增"/>
  <event id="[input] 帐号" name="setValue" value="{userAccount}"/>
  <event id="[input] 密码" name="setValue" value="{userPassword}"/>
  <event id="[input] 姓名" name="setValue" value="{userName}"/>
  <event id="[select] 性别" name="setText" value="{gender}"/>
  <event id="[date] 生日" name="setValue" value="{birthDate}"/>
  <event id="[select] 国籍" name="setText" value="{nationality}"/>
  <event id="[input] 描述信息" name="setValue" value="{description}"/>
  <event id="[button] 保存" wait="3"/>
  <event id="[gr... 帐号" name="assertExist"/>
</method>
```

基于UI控件封装模式的脚本

## 影响测试脚本学习、编写及维护成本的因素

✓ 系统架构影响：不同类型的被测系统需要依赖不同的底层测试框架，Selenium/Cypress/Appium/UIAutomation...

✓ 编码规范影响：前端编码不规范、控件缺少唯一ID、页面Frame嵌套、弹出新页面、嵌套非Web对象...

✓ UI架构影响：页面源码复杂、前端框架变更、样式布局变化...

## 自动化测试设计模式的核心是“隔离”和“复用”

通过自动化测试框架隔离测试脚本对页面源码及底层测试框架的依赖，通过封装和参数化提高测试脚本的复用度。

将底层技术和不稳定因素封装到自动化测试框架中，虽有效屏蔽，但并未真正消除。





# AI技术在自动化测试领域的应用





# 期望达到的目标

01

测试技术与  
系统架构  
无关

02

测试脚本与  
页面源码  
无关

03

测试脚本用  
自然语言  
零编码



# UI测试可用的AI技术

## 计算机视觉

基于模板匹配、特征点匹配算法的  
OpenCV类库及图像表征学习算法

## OCR识别

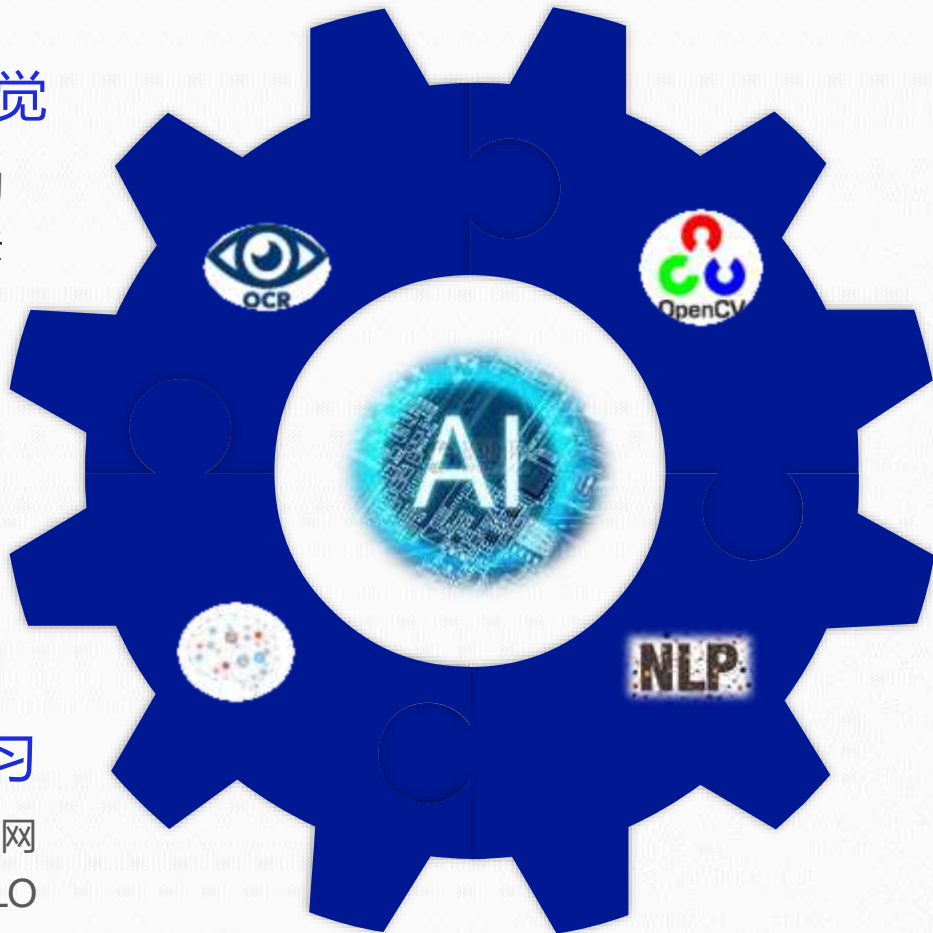
基于深度学习实现文本检测和识别的OCR（光学字符识别）技术

## 深度学习

适合小目标检测场景的卷积神经网络（CNN）模型YOLO

## 自然语言处理

领域特定语言（DSL）、词向量（Word2Vec）等NLP相关技术

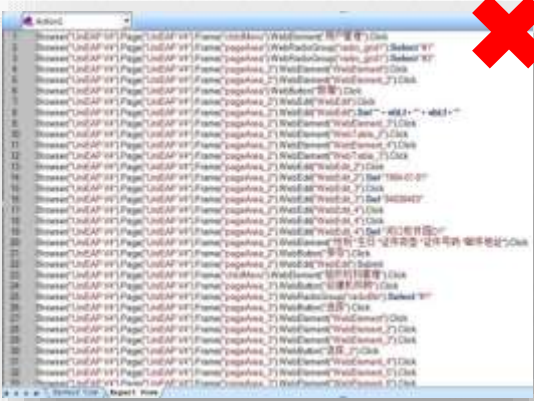




# 基于AI技术的UI自动化测试

利用OCR、图像识别、深度学习、自然语言处理等AI技术，让自动化测试可以同时满足“技术门槛低”和“脚本稳定性强”两个关键要求。

单纯脚本录制模式



- 上手看似容易
- 调试成本很高
- 脚本稳定性极差
- 无法大范围推广

对象封装/关键字驱动模式

```
<method id="创建用户">
  <event id="[menu]组织机构.用户管理"/>
  <event id="[button]新增"/>
  <event id="[input]账号" name="setValue" value="{userAccount}"/>
  <event id="[input]密码" name="setValue" value="{userPassword}"/>
  <event id="[input]姓名" name="setValue" value="{userName}"/>
  <event id="[select]性别" name="setText" value="{sex}"/>
  <event id="[date]生日" name="setValue" value="{birthday}"/>
  <event id="[select]国籍" name="setText" value="{nationality}"/>
  <event id="[input]描述信息" name="setValue" value="{description}"/>
  <event id="[button]保存" wait="3"/>
  <event id="[gridCell]userGrid,{userAccount},账号" name="assertExist"/>
</method>
```

项目结构		测试方法列表 【测试管理 > 测试计划查询】			
菜单页面		名称	耗时(秒)	启用	截图
非菜单页面		初始化菜单	2	是	是
搜索全部		计划查询	2	是	是

测试步骤列表				
类型	标识	操作	输入	
select	年份	setText	年份	
select	团队	setText	团队	
select	迭代	setText	迭代	

- 脚本稳定性较强
- 无法做到开箱即用
- 对前端开发规范性要求高
- 需要熟悉前端的开发参与

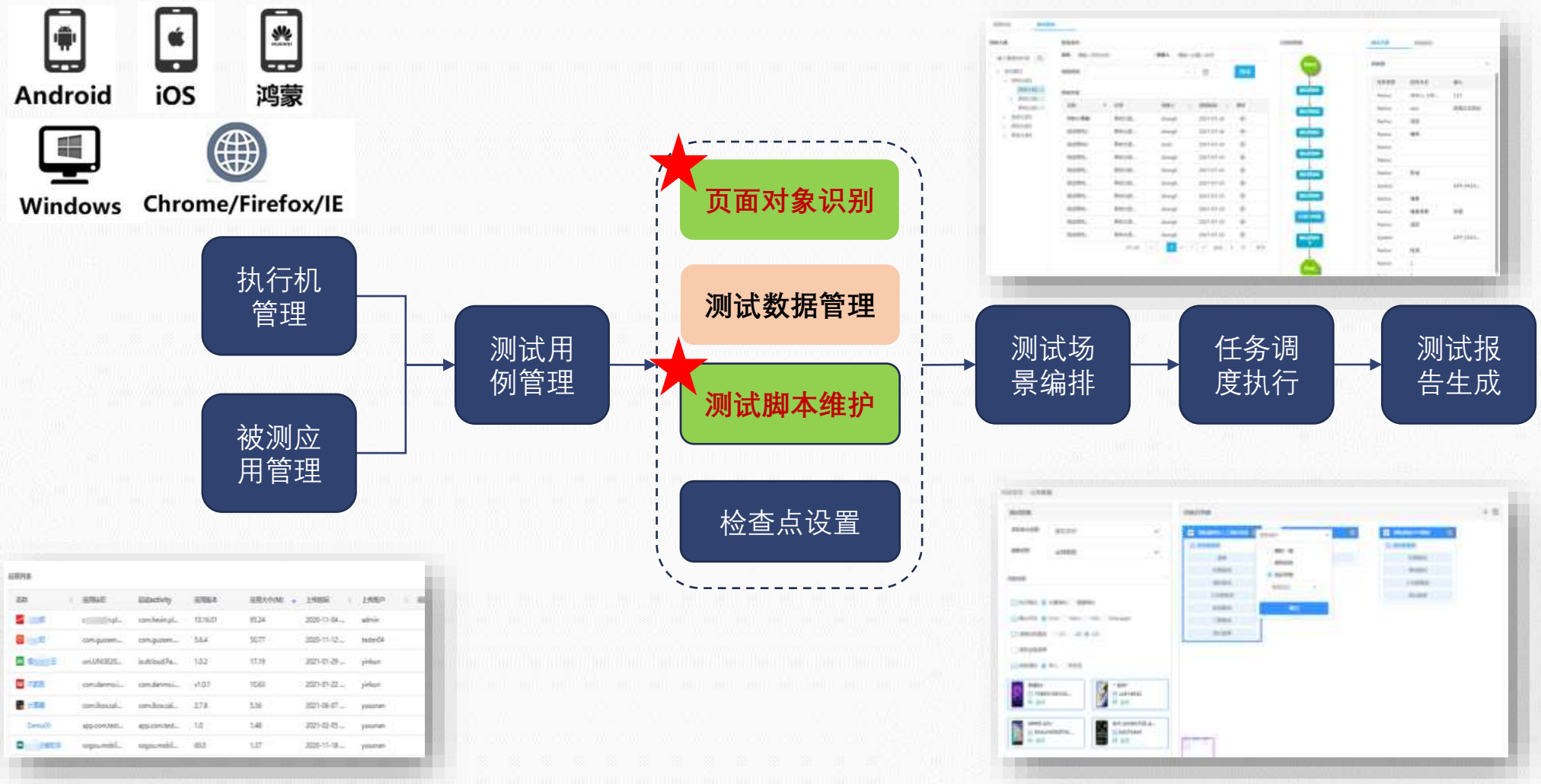
AI识别模式



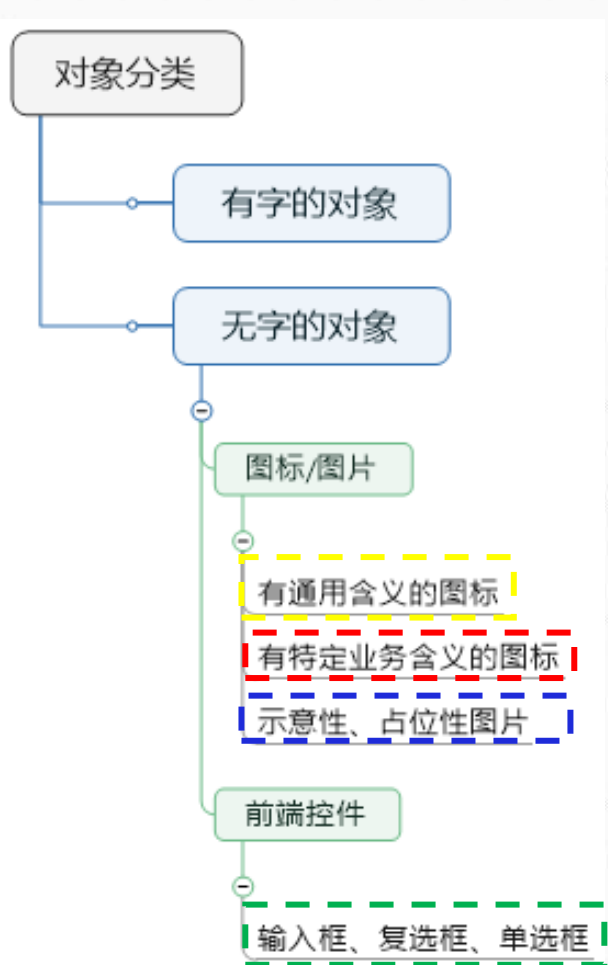
- 脚本稳定性较强
- 上手容易，不需要开发技能
- 基于AI技术屏蔽对被测系统技术架构及页面源码的依赖



# 需要用AI技术来解决什么问题？



# 基于视觉维度的页面对象分类





# 基于业务维度的测试数据参数化

❏ 页面对象的标识需要写入自动化测试脚本，测试数据需要通过参数化解耦后写入测试数据文件中。



新增日报

对应任务 日常管理、功能策划及市场推广

日期 2023-05-20

所用时间(人时) 6

产品特性策划讨论

工作内容描述

保存 保存并新增

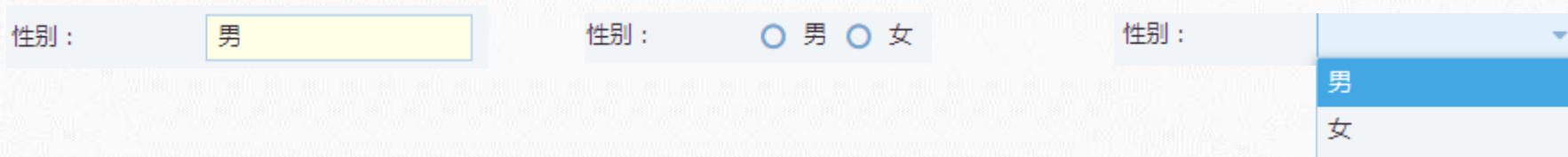
- 页面对象

“保存”按钮、“保存并新增”按钮、  
日期输入框、所用时间输入框、工作内容描述输入框

- 测试数据

“2023-05-20”、“6”、“产品特性策划讨论”

表面的区分原则：页面初始化之后就在页面上显示的是页面对象，后期输入的是测试数据



性别：男

性别： ☐ 男 ☐ 女

性别： 

男

女

应该基于业务维度判断哪些是测试数据，而不应该“键盘输入的就是数据，鼠标点击的就是对象”





# 基于业务维度的测试数据参数化

页面对象的标识需要写入自动化测试脚本，测试数据需要通过参数化解耦后写入测试数据文件中。

计费

按时计费

按月计费

\* 按时计费会在每个月底根据您的使用情况给您提供账单，需求提前充值；按月计费会在每个月底给您提供账单，以月为单位计费，请根据您的情况选择适合的计费方式。

地域

华南节点1

华南节点2

华北节点1

华南节点1

华南节点2

华北节点1

\* 地域代表您的主机所在的区域，选择距离您最近的区域有利于您的使用。  
\* 不同地域之间的产品内网不互通，且订购之后不支持更换地域，请您谨慎选择。

配置

1核

2核

4核

8核

16核

1G

2G

4G

8G

16G

32G

64G

```

<event type="云主机配置" id="计费方式" name="setValue" value="${计费方式}"/>
<event type="云主机配置" id="地域" name="setValue" value="${地域}"/>
<event type="云主机配置" id="CPU" name="setValue" value="${CPU核数}"/>
<event type="云主机配置" id="内存" name="setValue" value="${内存大小}"/>
    
```

	A	B	C	D
1	计费方式	地域	CPU核数	内存大小
2	按月计费	华南节点1	1核	1G
3	按月计费	华南节点2	2核	4G
4	按月计费	华北节点1	1核	1G
5	按月计费	华北节点1	2核	4G
6	按月计费	华南节点1	4核	8G
7	按月计费	华南节点2	4核	16G
8	按月计费	华北节点1	4核	32G
9	按月计费	华北节点1	4核	64G
10	按时计费	华南节点1	1核	1G
11	按时计费	华南节点2	2核	4G
12	按时计费	华北节点1	1核	1G
13	按时计费	华北节点1	2核	4G
14	按时计费	华南节点1	4核	8G
15	按时计费	华南节点2	4核	16G
16	按时计费	华北节点1	4核	32G
17	按时计费	华北节点1	4核	64G

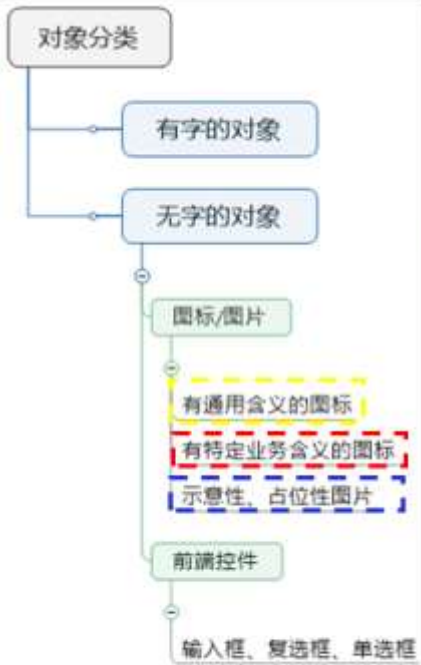
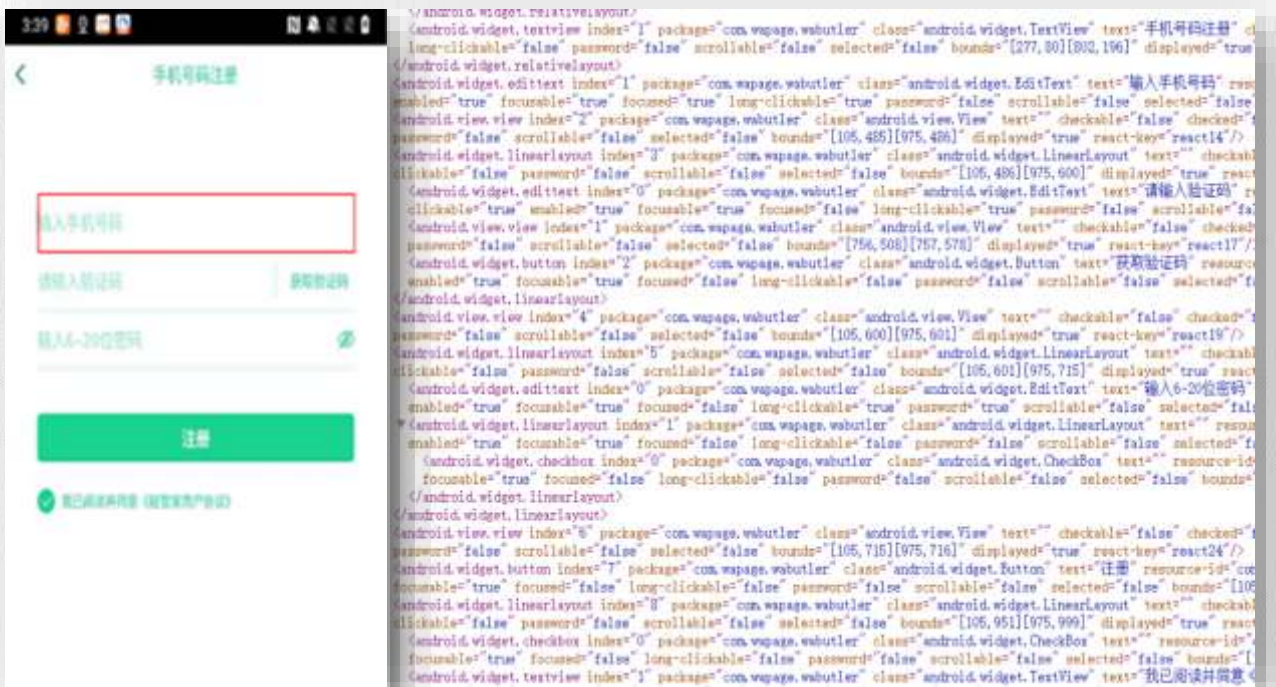
云主机配置





# 传统对象识别 VS AI对象识别

传统对象识别技术基于页面源代码，AI对象识别则颠覆性的摆脱了对页面源码的依赖，是UI自动化测试的趋势。



- 不同类型的被测系统需要依赖不同的底层测试框架  
Selenium/Cypress/Appium/UIAutomation...
- 对象识别率受编码规范及前端技术影响  
控件缺少唯一ID、页面Frame嵌套、弹出新页面、嵌套非Web对象...
- 对象维护工作量受源码复杂度、UI框架升级、样式变化等影响很大；

- 图像识别  
基于模板匹配、特征点匹配算法的OpenCV类库
- OCR识别  
基于深度学习实现文本检测和识别的OCR技术
- 深度学习  
适合小目标检测场景的卷积神经网络（CNN）模型YOLO







# 有字对象的AI识别——基于深度学习的OCR

基于文字定位对象是自动化测试常用的方法，利用OCR技术可以将文字直接转换为坐标，脱离对源码的依赖。对于重复文字对象可以通过索引或在局部图片中进行查找的方式进行准确定位。



基于文字定位页面对象位置



指定重复对象的索引顺序  
(适用于位置固定)



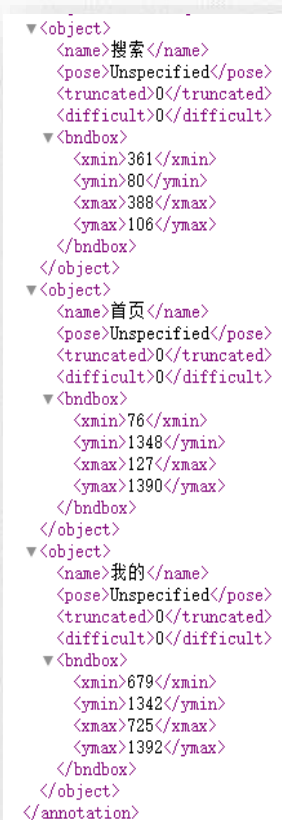
在指定局部图片中定位目标  
(适用于位置不固定)





# 无字对象的AI识别——通用图标的深度学习训练识别

- 大部分应用的图标都具备一定的相似性（登录、搜索、购物车、个人中心等），可以利用计算机视觉，结合深度学习相关技术实现自动化定位。



确定通用图标范围



准备标注图片库



人工标注

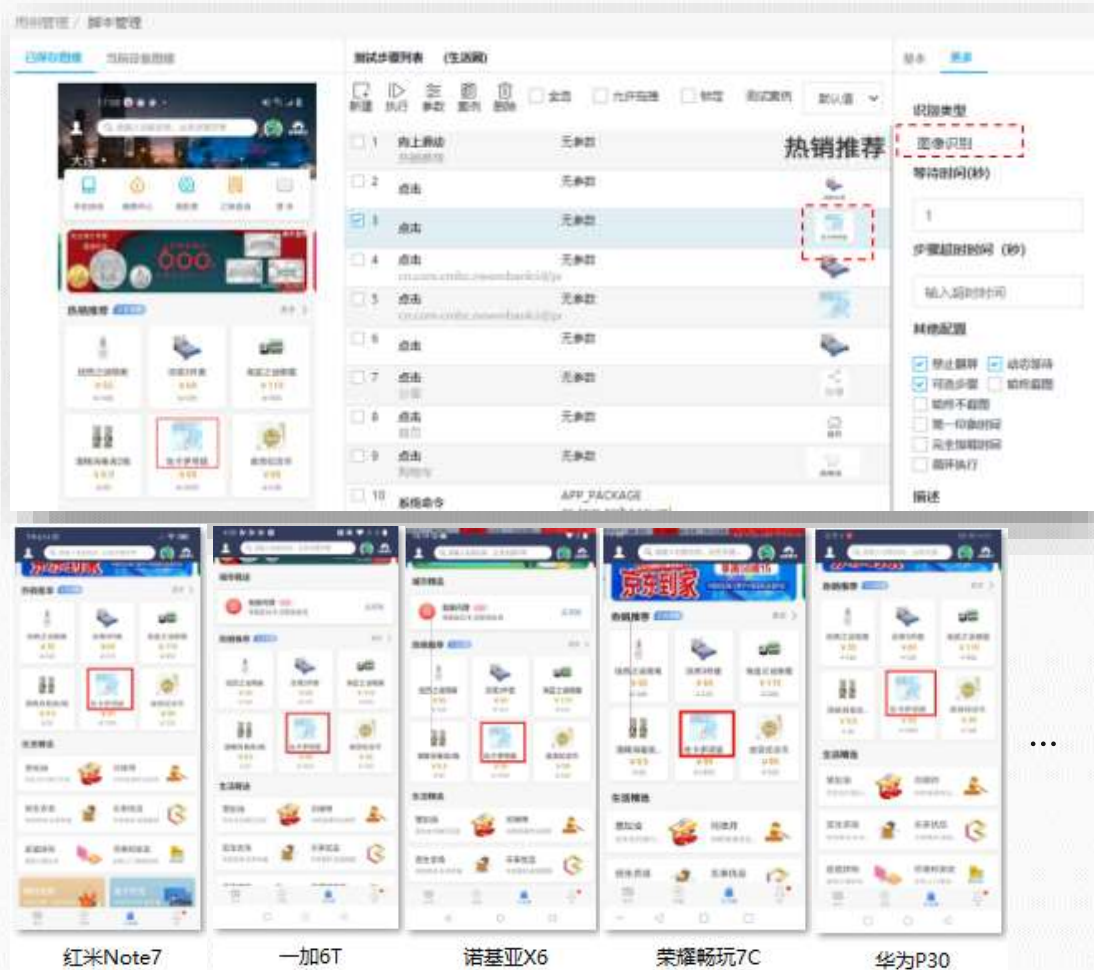


训练



# 无字对象的AI识别—非通用图片的人工标注识别

- 业务含义不明显、不常操作、经常更换、特征不明显的图片可以直接通过图像识别的方式进行定位操作。单步骤使用时可以直接截图操作，如果考虑多步骤多用例复用可以给截图设置标签。





# 自然语言测试脚本语法设计

基于可文本化描述的目标定位对象、参照关键字、方位、索引、偏移像素适应各种场景的定位描述语法。

分类	语法结构	语法示例
鼠标操作	操作类型 + 索引（可选） + 目标定位对象	点击 第 2 个 XXX
	操作类型 + 索引（可选） + 参照关键字 + 方位 + 索引（可选） + 目标定位对象	点击 第 1 个 输入框
	操作类型 + 索引（可选） + 参照关键字 + 方位 + 索引（可选） + 目标定位对象	点击 第 2 个 YYY 右侧 第 1 个 XXX
	操作类型 + 索引（可选） + 参照关键字 + 方位 + 像素	点击 第 2 个 YYY 右侧 第 30 个像素
键盘操作	输入 指定文字/@参数名	输入 张三 输入 @姓名
	向 + 索引（可选） + 参照关键字 + 方位（可选，默认向右） + 像素（可选，默认50像素） + 输入 + 指定文字/@参数名	向 第2个 价格 输入 500元 向 价格 输入 @商品单价 向 价格 下侧 第100个像素 输入 500元
	向 + 索引（可选） + 目标定位对象（HTML控件） + 输入 + 指定文字/@参数名	向 第2个 输入框 输入 500元
	输入快捷键 快捷键/组合快捷键/向上/向下/向左/向右	输入快捷键 ctrl 输入快捷键 ctrl+enter 输入快捷键 向左
数据处理	@参数名=固定值	@参数名=张三
	@参数名=参照关键字 + 方位 + 索引 + 目标定位对象（文本）	@参数名=张三 右侧 第2个 文本
检查点	检查 + 存在/不存在 固定值	检查 存在/不存在 XXX
	检查 + 参照关键字 + 方位 + 索引 + 目标定位对象（文本） + 检查类型 + 指定文字/@参数名	检查 余额 右侧 第 1 个 文本 等于 500
		检查 余额 右侧 第 1 个 文本 等于 @实际余额

目标定位对象	指定文本
	HTML控件（输入框、复选框、单选框）
	文本（OCR识别出来的任意内容）
参照关键字	YYY
方位	上/下/左/右 侧
索引	第n个（从左到右、从上到下）
	倒数第n个（从下到上、从右到左）
像素	第 n 个像素
事件	鼠标操作：点击、双击、右键、移到…
	屏幕操作：点击、长按…
	键盘操作：输入、输入快捷键…
	检查操作：存在/不存在/等于/不等于/包含/不包含/大于/小于/不大于/不小于

点击 返回

点击 全部应用

点击 大牌直降 右侧 的 进入

点击 第2个 全部

点击第2个全部 右侧30像素





# 自然语言测试脚本编辑器

基于设计好的脚本语法，提供脚本编写辅助编辑器，支持语法提示、语法校验、在线调试等能力。

用例管理 / 脚本管理

已保存用例 当前设备图像

华为 P30

对象模式 41分28秒 验证

测试步骤列表 (演示)

新建 执行 按测试 参数 案例 删除 复制 标注+识别

☐ 全选 ☐ 允许拖拽 ☐ 锁定 测试案例 默认值

序号	操作	参数	备注
1	点击存款	无参数	点击存款
2	点击我的	无参数	点击我的
3	输入	AutoVarName1629279898702 用户名	输入用户名
4	输入	AutoVarName1629279903936 密码	输入密码

请输入AI步骤,如点击,输入,滑动等, shift+enter执行

点击 关键字  
输入 参数,或@引用参数  
点击第 n 个 关键字  
返回  
手势密码 九宫格密码图形对应的数字,直接输入数字或@引用参数  
点击 关键字上/下/左/右 方第 n 个 关键字  
长按 关键字  
返回主屏幕  
如果存在 对象  
条件分支

基本 更多

文本

坐标  
[0,0][0,0]  
定位策略  
☒ AI定位  
交互类型

操作

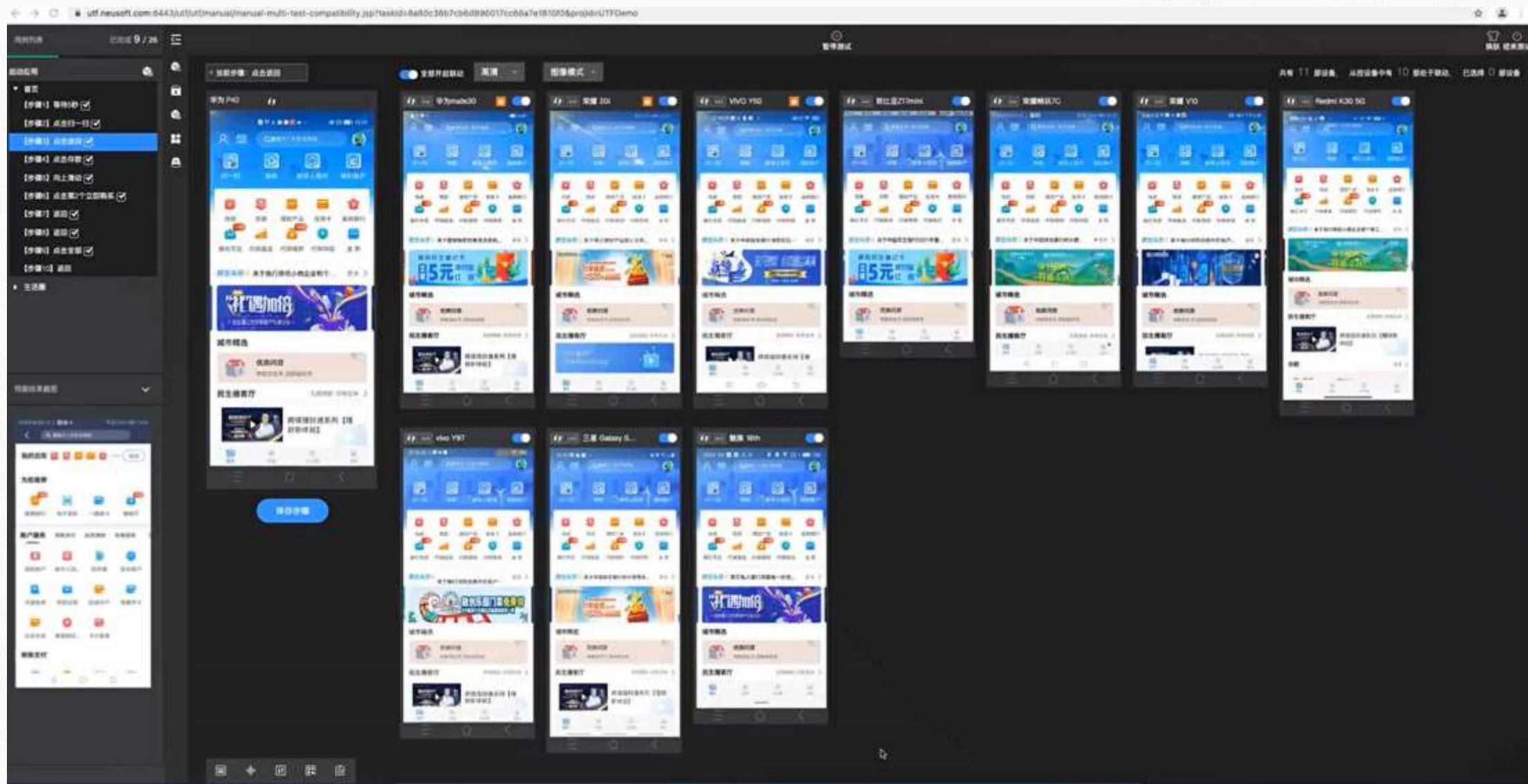
输入 参数,或@引用参数  
点击第 n 个 复选框/输入框/单选框  
向第 n 个 关键字上/下/左/右/侧输入 参数,或@引用参数  
向 关键字上/下/左/右/侧输入 参数,或@引用参数  
向 关键字输入 参数,或@引用参数

输入

保存

# 基于智能图像识别的跨终端群控测试

基于智能图像识别技术也可以实现手工批量兼容性测试，在功能可以稳定回归测试之前，大幅提升多终端兼容性测试效率。





# 自动化测试推广实施实践





# 什么样的项目适合做自动化测试?

## 技术参考指标

项目周期长、需要长期维护、版本发布频繁、  
质量要求高、采用敏捷开发模式、  
组织推行DevOps.....

## 关键非技术指标

- ❑ 当前测试覆盖度、质量风险及测试投入;
- ❑ 未来测试覆盖度提升目标、质量风险降低目标, 以及如果不引入自动化手段达成这些目标需要的人力投入;
- ❑ 这个目标是谁制定的或者是向谁承诺要达成的;



# AI自动化测试可以解决的问题

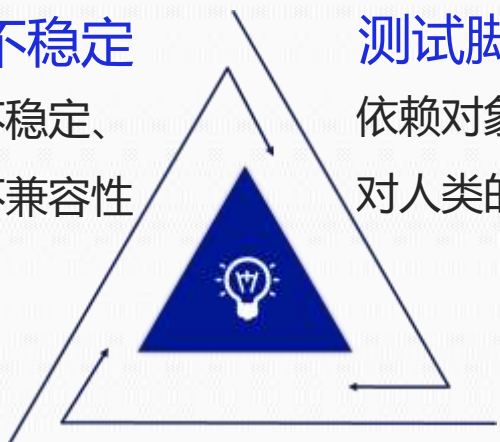
## 解决问题

### 对象定位不稳定

对象不识别、执行不稳定、  
跨终端不兼容性

### 测试脚本可读性差

依赖对象源码级的属性，  
对人类的可读性差



### 测试技术不统一

Web、Android、iOS、CS等应用底层  
的自动化测试技术各不相同

## 改进效果

- ❑ 主要使用页面文字和图标，脚本编写效率大幅提升；
- ❑ 基于自然语言的脚本，可读性大幅提升，适合手自一体化；
- ❑ 脚本调试所见即所得，无需依赖页面源码；
- ❑ UI框架/样式/布局升级变化对脚本影响较小；
- ❑ 无需解析DOM，可以借助GPU算力提高执行速度；
- ❑ 测试人员不用学习各种底层技术，可专注测试业务；





# 自动化脚本设计最佳实践

- ❑ 从业务视角区分什么页面对象（定义在脚本中）、什么是测试数据（参数化处理）；
- ❑ 每个测试用例的自动化脚本是一个可复用单元，步骤数量通常在30步骤左右；
- ❑ 测试用例要被编排到测试场景中才能被执行（培养用户模块化设计的习惯）；
- ❑ 保证测试场景是可独立执行的原子单元，互不依赖，可以并行、可以串行；
- ❑ 每个测试场景首先调用初始用例（如，登录、进入首页等）；
- ❑ 每个场景结尾调用回初始页面的用例（如，退出、回首页、重启应用、清缓存等）；





# 自动化测试ROI分析



## 降低质量风险

利用自动化提高回归测试的用例覆盖度、数据覆盖度以及回归频率，有效降低上线质量风险、



## 提高测试效率

通过7\*24小时无人值守的方式提高测试效率，加快测试反馈速度，确保持续集成的时效性、提高项目整体敏捷度。



## 降低回归成本

对于已有回归比较频繁的部分测试任务，通过自动化测试可以降低执行成本、解放人力。





提问时间

谢谢大家





关注社区公众号  
了解更多活动

