Яндекс

Гейзенбаг

Агентные системы для мобильного регресса

Как сделать инструмент выполнения сценариев в мобильных

Артем Зощук

Руководитель группы инфраструктуры автоматизации тестирования.

Яндекс, Персональные сервисы.

01 Что такое агентная система

02 Проблема, задача, критерии успеха

03 Граф агентной системы

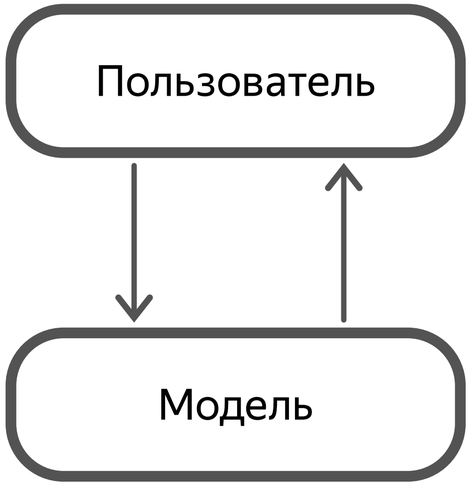
04 Инструменты агентной системы

01

Что такое агентная система

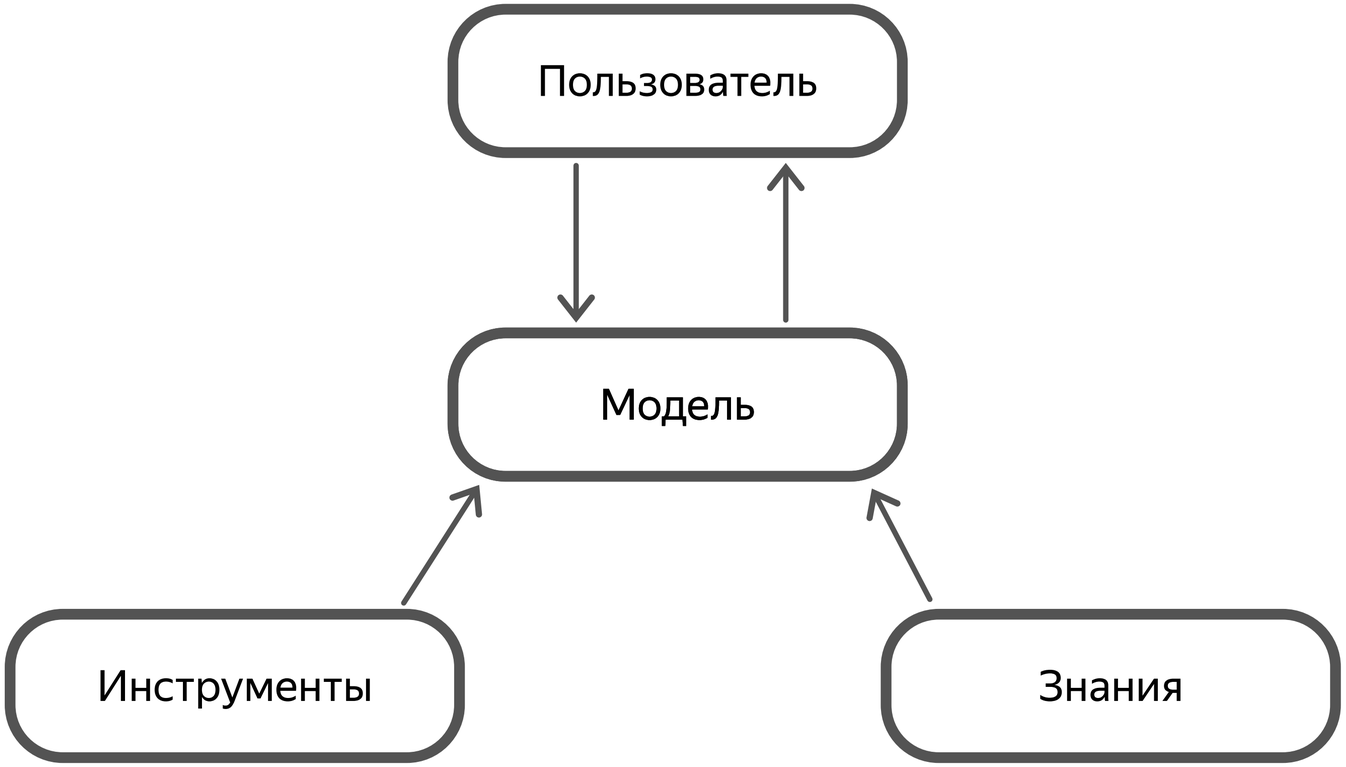
Кто такие эти ваши агентные системы

Работа с моделью



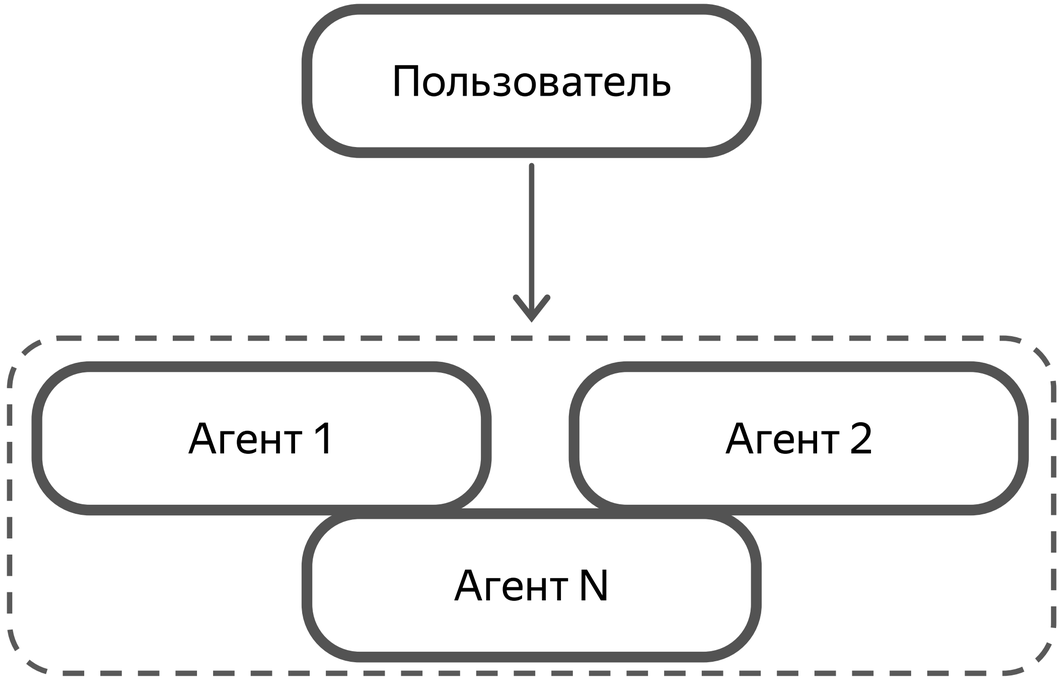
Модель полагается на информацию от пользователя

Агент



Агент может изменять внешний мир и получать из него данные

Агентная система



Система,
объединяющая больше
одного агента

02

Проблема, задача, критерии успеха

Зачем начали строить систему и какую поставили задачу

Зачем

01

Не успеваем автоматизировать сразу

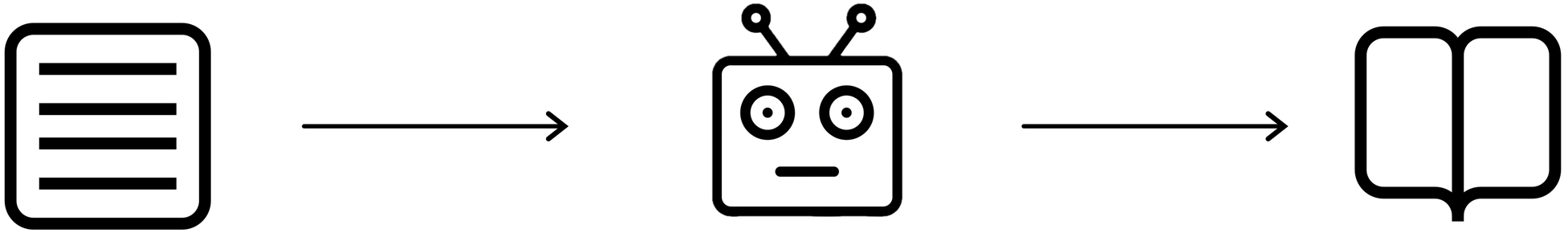
02

Автотесты не всегда оправданы

03

Хотим уметь предтестировать задачу

Что хотим получить



Сценарии

Железный тестировщик

Отчёт

Сценарий

Предусловия

1 Авторизоваться пользователем с подпиской Амедиатека + START

Шаги

1

Перейти на таб "Главное"

Пролистать до "Витрины стримингов"

В блоке "Витрины стримингов" тапнуть на любую точку входа

> Скрин

Ожидаемый результат

- Произошел переход на соответствующую контентную витрину
- В промоблоке есть кнопка "Неинтересно", выглядит как белый круг с минусом внутри

> Скрин

2

Нажать на кнопку "Неинтересно"

Ожидаемый результат

- Проигрывается анимация кнопки
- Кнопка неинтересно меняет цвет с белого на красный

Отчёт

Test body

- Прекондишн 1: Авторизоваться пользователем с подпиской Амедиатека \+ START 22 sub-steps, 1 attachment
- Шаг 1: Перейти на таб "Главное" Пролистать до "Витрины стримингов" В блоке "Витрины стримингов" тапнуть на любую точку входа 14 sub-steps
- Ожидание 1: * Произошел переход на соответствующую контентную витрину * В промоблоке есть кнопка "Неинтересно", выглядит как белый круг с минусом внутри
  3 parameters, 1 attachment
- Шаг 2: Нажать на кнопку "Неинтересно" 23 sub-steps
- Ожидание 2: * Проигрывается анимация кнопки * Кнопка неинтересно меняет цвет с белого на красный 3 parameters, 1 attachment

Ожидаемый результат	* Проигрывается анимация кнопки * Кнопка неинтересно меняет цвет с белого на красный
Фактический результат	Successfully executed the action by clicking on the 'Неинтересно' button. The VLM analysis confirms that the button animation played and the color changed from white to red/orange as expected. The button is now visually highlighted in a reddish-orange color, indicating the 'not interested' state has been activated. Both expectations are fully met.
Резолюция	pass

- Screenshot Step 2 — 890.3 KiB
- LLM Trace — 851.2 KiB
- State Dump — 110.5 KiB
- Video Recording — 20.8 MiB


P: 0 / 0 X: 0.0 Y: 0.0 Xv: 0.0 Yv: 0.0 Prs: 0.0 Size: 0.0
10:17
Wed, Apr 8

Как двигаемся

Как двигаемся

Метрики качества

Пропущенные баги

Самый дорогой тип ошибок

Мнимые баги

Допустимы в разумных количествах

Воспроизводимость

Стабильность резолюций

Бенчмарк

Сценарии без багов

Считаем мнимые баги

Сценарии с заложенными багами

Считаем пропущенные баги

Заранее определитесь с тем, что вы будете считать успехом

03

Граф агентной системы

Строим процесс и распределяем роли

Фреймворки

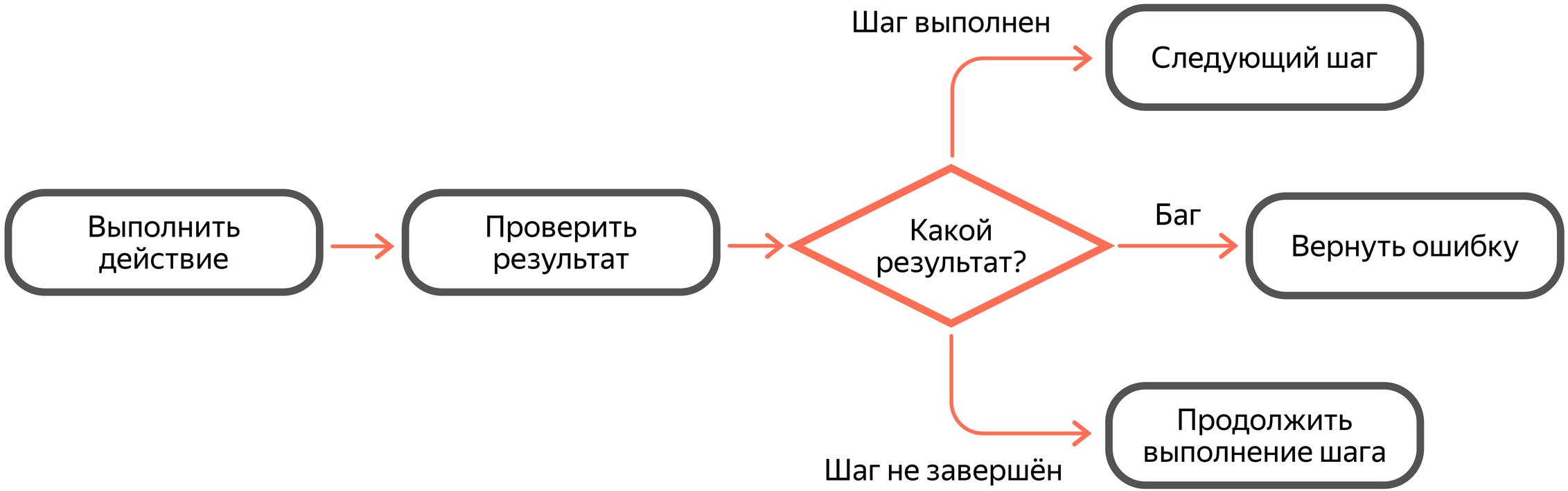
Langchain

- Агенты
- Взаимодействие с LLM

Langgraph

- Агентные системы

Выполнить шаг тестового сценария



Лень

Решение

Отдать перепланирование другому агенту

Много контекста

Решение

- Сжатие старых сообщений
- Оптимизация формата ответа

Унесённые призраками (2001), Хаяо Миядзаки.

Стройте самый простой флоу, который может решить вашу задачу

Dump

```
<node index="1" text=""
      resource-id="ru.kinopoisk:id/video_feed"
      class="android.widget.FrameLayout"
      package="ru.kinopoisk" content-desc="Что смотреть"
      checkable="false" checked="false"
      clickable="true" enabled="true"
      focusable="true" focused="false"
      scrollable="false" long-clickable="false"
      password="false" selected="false"
      bounds="[216,2060][432,2214]"/>
```

Dump

```
<node index="1" text=""
      resource-id="ru.kinopoisk:id/video_feed"
      class="android.widget.FrameLayout"
      package="ru.kinopoisk" content-desc="Что смотреть"
      checkable="false" checked="false"
      clickable="true" enabled="true"
      focusable="true" focused="false"
      scrollable="false" long-clickable="false"
      password="false" selected="false"
      bounds="[216,2060][432,2214]"/>
```

JSON


```
{
  "text": "",
  "resource-id": "ru.kinopoisk:id/video_feed",
  "class": "android.widget.FrameLayout",
  "content-desc": "Что смотреть",
  "checkable": false, "checked": false,
  "enabled": true,
  "focused": false,
  "scrollable": false,
  "selected": false,
  "bounds": "[ 216,2060 ][ 432,2214 ]"
}
```

JSON


```
{
  "text": "",
  "resource-id": "ru.kinopoisk:id/video_feed",
  "class": "android.widget.FrameLayout",
  "content-desc": "Что смотреть",
  "checkable": false, "checked": false,
  "enabled": true,
  "focused": false,
  "scrollable": false,
  "selected": false,
  "bounds": "[ 216,2060 ][ 432,2214 ]"
}
```

TOONish

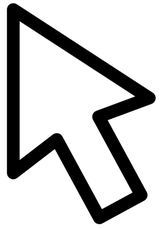
```
FrameLayout [ru.kinopoisk:id/video_feed]:
    "" "Что смотреть" ( 216,2060, 432,2214 ) {enabled}
```

04

Инструменты агентной системы

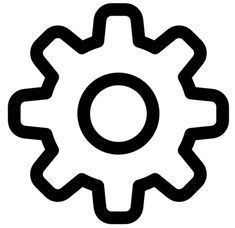
Чем потрогать приложение

Какое взаимодействие нужно



Действия в приложении

Состояние приложения



Настройки устройства

Простой уровень

adb shell input

```
# тап по координатам X Y:
adb shell input tap 500 1450

# свайп X1 Y1 X2 Y2 [duration(ms)]:
adb shell input swipe 100 500 100 1450 100

# ввод текста:
adb shell input text "text"

# нажатие кнопки:
adb shell input keyevent 82
```

Проблема

Не весь текст вводится

```
# ввод текста на английском:
adb shell input text "Hello world"
> success!
```

```
# ввод текста на русском:
adb shell input text "Привет, мир!"
> fail :(
```

Работа с системой

Broadcast Receivers

```
class ClipboardBroadcastReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        val clipboard = context.getSystemService(ClipboardManager::class.java)

        when (intent.action) {
            "mcp.clipboard.set" → {
                val text = intent.getStringExtra("text")
                val encoding = intent.getStringExtra("encoding")

                val decoded = if (encoding == "base64") {
                    String(Base64.decode(text, Base64.DEFAULT))
                } else text

                clipboard.setPrimaryClip(ClipData.newPlainText("mcp", decoded))
            }
            "mcp.clipboard.clear" → {
                clipboard.clearPrimaryClip()
            }
        }
    }
}
```

Пример вызова

```
adb shell am broadcast \
    -a mcp.clipboard.set \
    -e encoding base64 \
    -e text 0J/RgNC40LLQtdGCLCDQvNC40YAh \
    -n yandex.mcp.device.tool/.ClipboardBroadcastReceiver
```

UIAutomator

Instrumentation

```
class TextInputCleaner : Instrumentation() {
    override fun onCreate(arguments: Bundle?) {
        super.onCreate(arguments)
        start()
    }

    override fun onStart() {
        val device = UiDevice.getInstance(this)

        // Находим активное текстовое поле
        val textField = device.findObject(
        UiSelector().className("android.widget.EditText").focused(true)
        )

        // Очищаем текст
        if (textField.exists()) {
            textField.setText("")
        }

        finish(Activity.RESULT_OK, Bundle())
    }
}
```

Пример вызова

```
adb shell am instrument \
    -w yandex.mcp.device.tool/.TextInputCleaner
```

Как отдать инструменты агентной системе

01

Хорошее описание

02

Только то, что необходимо

03

Минимизировать шанс ошибки

MCP сервер

01

Удобно работать

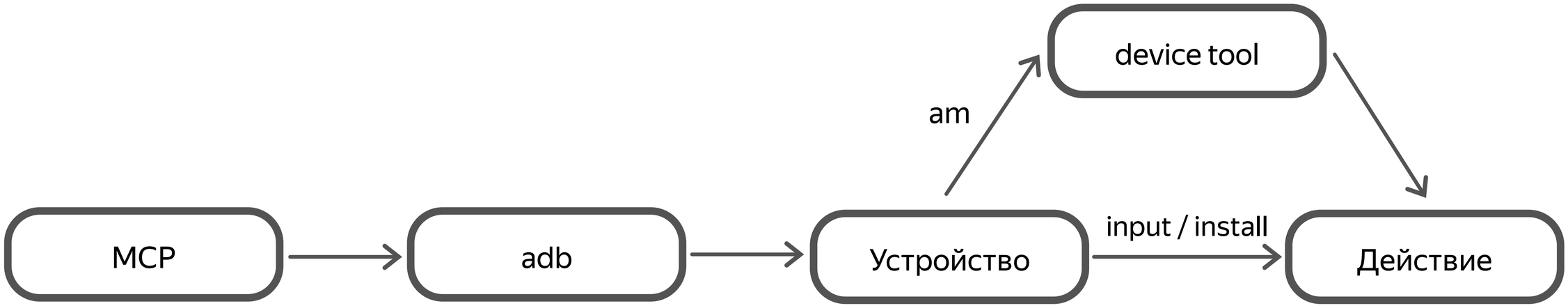
02

Можно переиспользовать

03

Хорошо поддерживается

Схема работы



Существующее решение

Mobile MCP

Почему не используем

01

Описание инструментов и формат ответа

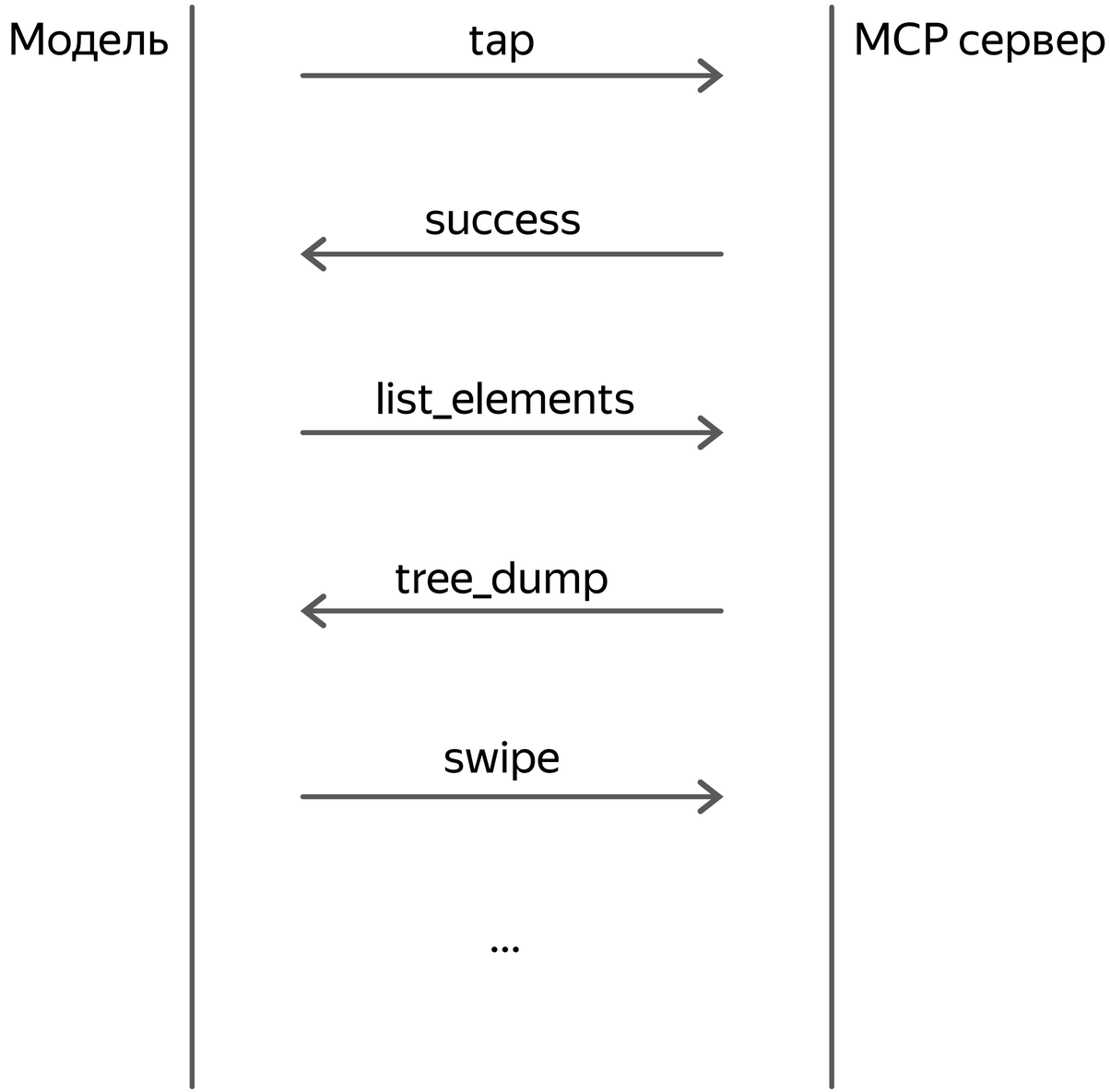
02

Внутренние инструменты

03

Много отладки

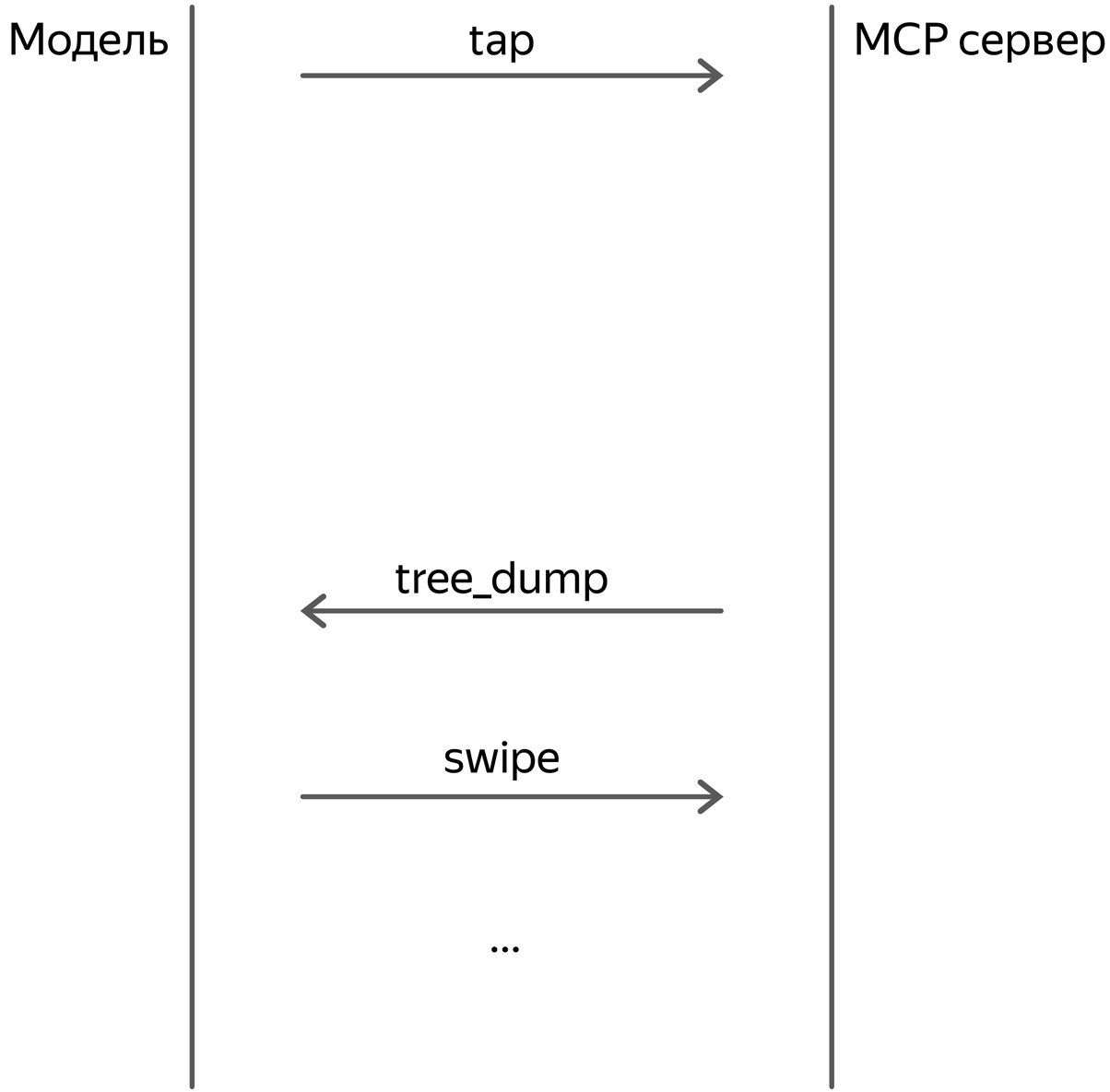
Много итераций



Много итераций

Решение

Сразу возвращать состояние приложения



Передача инструментов MCP в агента

```python
import asyncio
from langchain_mcp_adapters.client import MultiServerMCPClient
from langchain.agents import create_agent
from langchain_openai import ChatOpenAI

async def main():
    client = MultiServerMCPClient(
        {
            "mobile-mcp": {
                "transport": "stdio",
                "command": "node",
                "args": [ "/path/to/mcp/lib/index.js"],
            }
        }
    )

    all_tools = await client.get_tools()
    ALLOWED_TOOLS = ["take_screenshot", "tap", "swipe"]
    tools = [t for t in all_tools if t.name in ALLOWED_TOOLS]

    mobile_agent = create_agent(model, tools=tools)
```

Передача инструментов MCP в агента

```
import asyncio
from langchain_mcp_adapters.client import MultiServerMCPClient
from langchain.agents import create_agent
from langchain_openai import ChatOpenAI

async def main():
    client = MultiServerMCPClient(
        {
            "mobile-mcp": {
                "transport": "stdio",
                "command": "node",
                "args": [ "/path/to/mcp/lib/index.js"],
            }
        }
    )

    all_tools = await client.get_tools()
    ALLOWED_TOOLS = ["take_screenshot", "tap", "swipe"]
    tools = [t for t in all_tools if t.name in ALLOWED_TOOLS]

    mobile_agent = create_agent(model, tools=tools)
```

Передача инструментов MCP в агента

```
import asyncio
from langchain_mcp_adapters.client import MultiServerMCPClient
from langchain.agents import create_agent
from langchain_openai import ChatOpenAI

async def main():
    client = MultiServerMCPClient(
        {
            "mobile-mcp": {
                "transport": "stdio",
                "command": "node",
                "args": [ "/path/to/mcp/lib/index.js"],
            }
        }
    )

    all_tools = await client.get_tools()
    ALLOWED_TOOLS = ["take_screenshot", "tap", "swipe"]
    tools = [t for t in all_tools if t.name in ALLOWED_TOOLS]

    mobile_agent = create_agent(model, tools=tools)
```

Передача инструментов MCP в агента

```python
import asyncio
from langchain_mcp_adapters.client import MultiServerMCPClient
from langchain.agents import create_agent
from langchain_openai import ChatOpenAI

async def main():
    client = MultiServerMCPClient(
        {
            "mobile-mcp": {
                "transport": "stdio",
                "command": "node",
                "args": [ "/path/to/mcp/lib/index.js"],
            }
        }
    )

    all_tools = await client.get_tools()
    ALLOWED_TOOLS = ["take_screenshot", "tap", "swipe"]
    tools = [t for t in all_tools if t.name in ALLOWED_TOOLS]

    mobile_agent = create_agent(model, tools=tools)
```

Выводы

01

Образ результата

Как поймём успех.

Как должно выглядеть.

02

Описание процесса

От простого к сложному.

Как не перегрузить модель.

03

Дизайн инструментов

Реализация важнее технологии.

Ограничений на взаимодействие нет.

Яндекс

Спасибо

Артем Зощук

Руководитель группы инфраструктуры автоматизации тестирования.

Яндекс, Персональные сервисы.