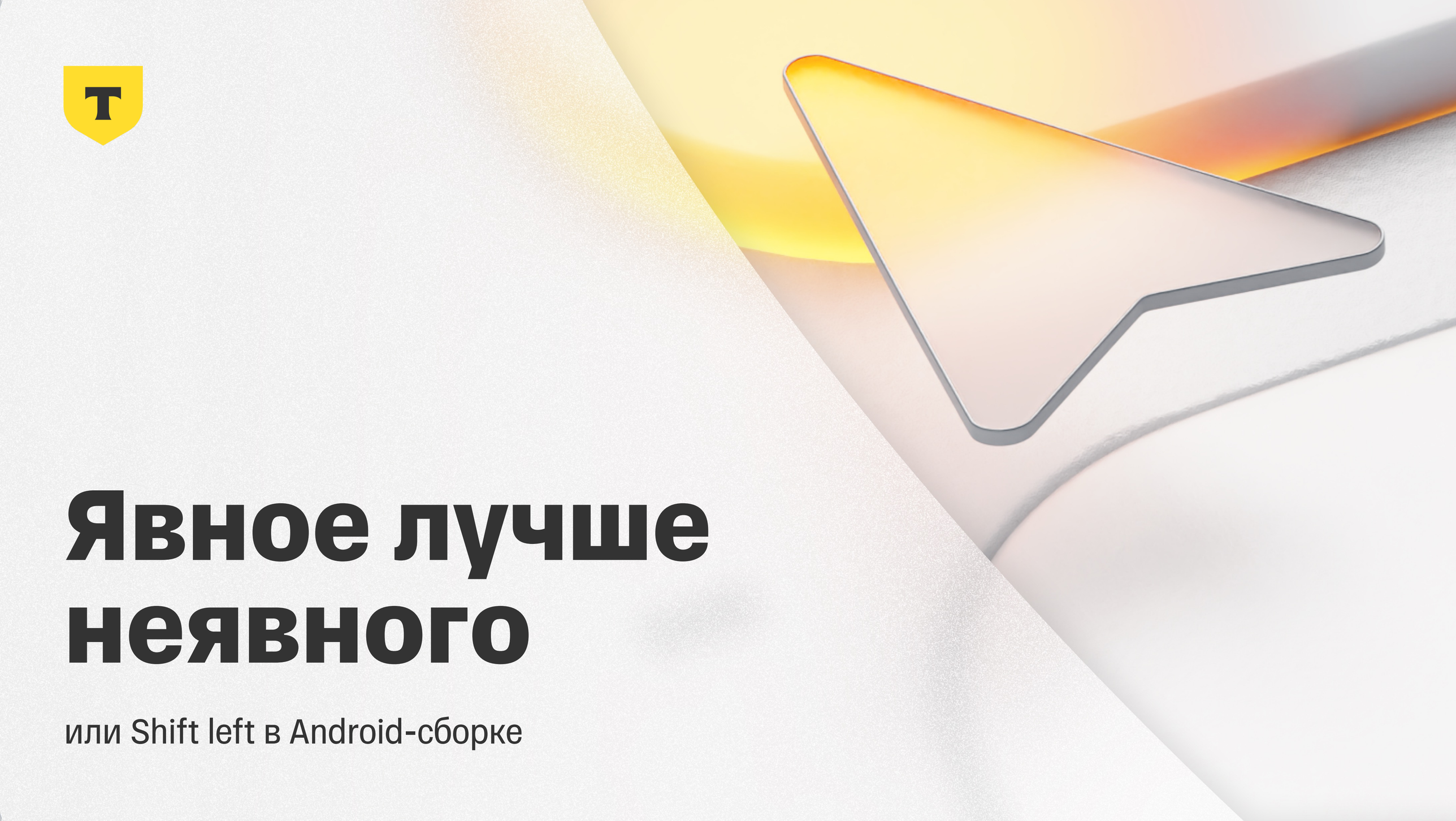


Явное лучше неявного

или Shift left в Android-сборке

Юрий Анисимов

Staff-инженер

- Разрабатываю приложение для физлиц
- В Т-Банк почти 6 лет
- Занимаюсь сборкой, автоматизациями,...

Собирать быстро и без сбоев

Мобильный банк в цифрах

150+ команд

Участвуют в разработке Мобильного банка

100+ мерж коммитов

В среднем в день разработчики вливают в master ветку

600+ библиотек

Количество внешних зависимостей приложения

Собирать быстро и без сбоев

Мобильный банк в цифрах

150+ команд

Участвуют в разработке Мобильного банка

100+ мерж коммитов

В среднем в день разработчики вливают в master ветку

600+ библиотек

Количество внешних зависимостей приложения

2000+ внутренних

Семь раз подумайте, перед тем как «выезжать» из репозитория

Доклад о стабильности сборки

Этапы процесса разработки

Разработка

Локальный запуск
на машине разработчика

Интеграция

Запуск в пайплайне
merge (pull) request

Деплой

После попадания кода
в master ветку

Этапы процесса разработки

Разработка

Локальный запуск
на машине разработчика

Интеграция

Запуск в пайплайне
merge (pull) request

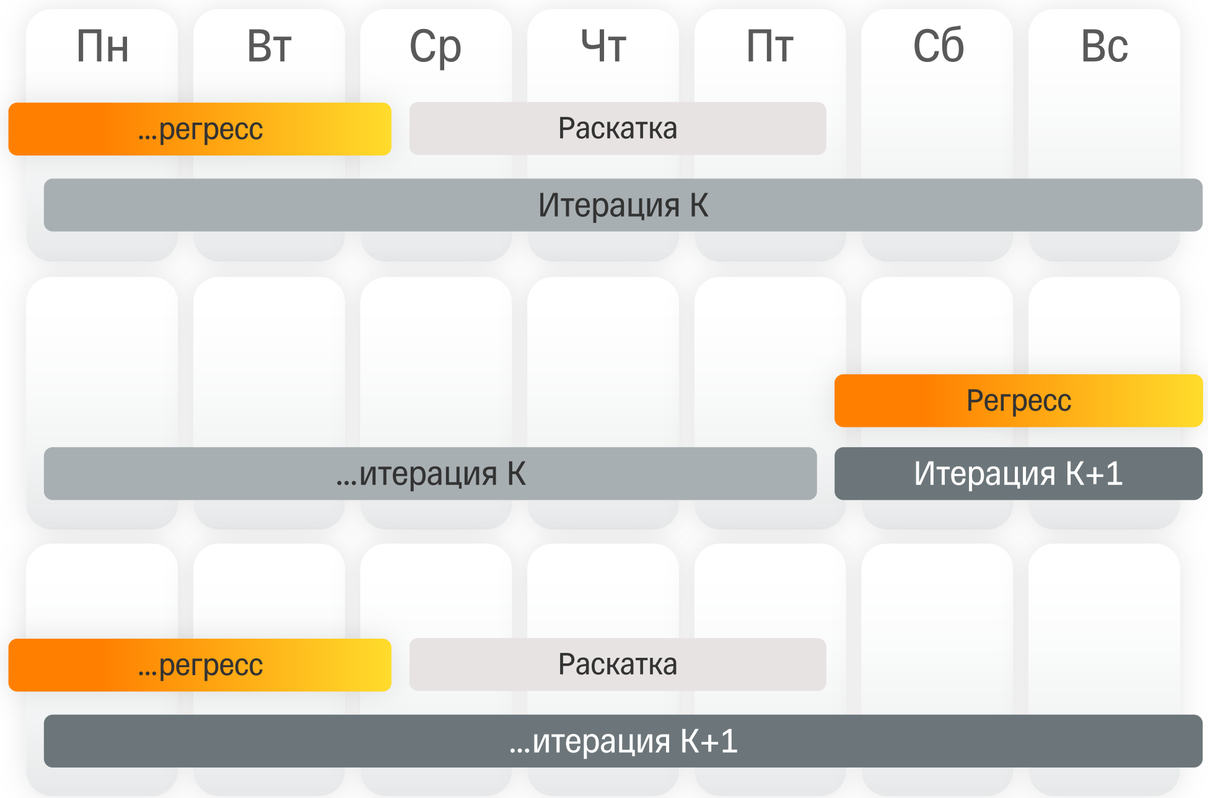
Деплой

После попадания кода
в master ветку

Будни платформенной команды

Agile Release Trains

- Релизы по расписанию
- Синхронизация всех команд



Будни платформенной команды

Реальный случай

из нашей практики

Нам отклонили обновление

APK содержит DLL файлы

Будни платформенной команды

Реальный случай

из нашей практики

Нам отклонили обновление

APK содержит DLL файлы

Какие DLL файлы? Откуда?

Будни платформенной команды

Реальный случай

из нашей практики

Нам отклонили обновление

APK содержит DLL файлы

Какие DLL файлы? Откуда?

Нет времени разбираться!

Назначен старт рекламной компании

Поиск обходного решения

Чтобы получить время на исправление

01 02 03 04

Гнев, отрицание

Меньше всего хочется
исправлять проблему
с «горящим задом»

Поиск обходного решения

Чтобы получить время на исправление

01

02

03

04

Гнев, отрицание

Меньше всего хочется
исправлять проблему
с «горящим задом»

Торг

Однако, мы знаем,
что DDL файлы не нужны
Андроид приложению

Поиск обходного решения

Чтобы получить время на исправление

01

Гнев, отрицание

Меньше всего хочется исправлять проблему с «горящим задом»

02

Торг

Однако, мы знаем, что DDL файлы не нужны Андроид приложению

03

Депрессия

По содержимому APK файла мы не можем установить источник

04

Поиск обходного решения

Чтобы получить время на исправление

01

Гнев, отрицание

Меньше всего хочется исправлять проблему с «горящим задом»

02

Торг

Однако, мы знаем, что DDL файлы не нужны Андроид приложению

03

Депрессия

По содержимому APK файла мы не можем установить источник

04

Принятие

Есть способ быстро пофиксить проблему, добавив в исключения

Суть проблемы

Позднее обнаружение

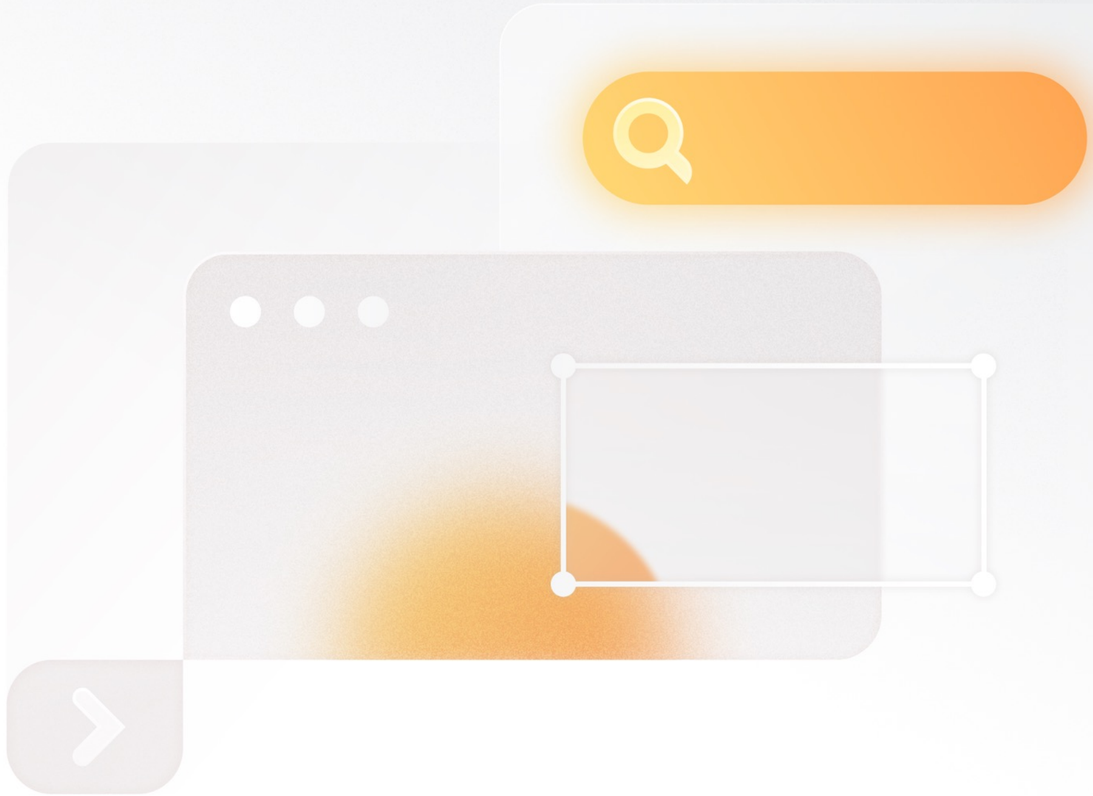
Обнаружили проблему на этапе публикации в магазин приложений

Отсутствие контроля

Изменения без чьего-либо ведома попали в приложение

А что еще может незаметно попасть в приложение?

План доклада



01 Поиск причины и источника

02 Концепция Shift-left для сборки

03 Возможности Gradle и AGP

04 Как неявное сделать явным

05 Бонус

01

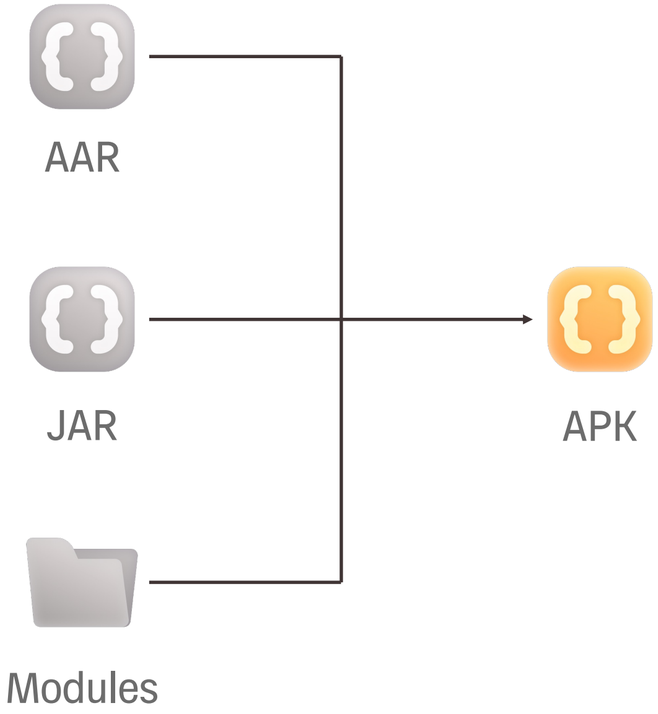
Поиск причины и источника

Причины ошибок после влития в master

01 Особенности сборки APK файла

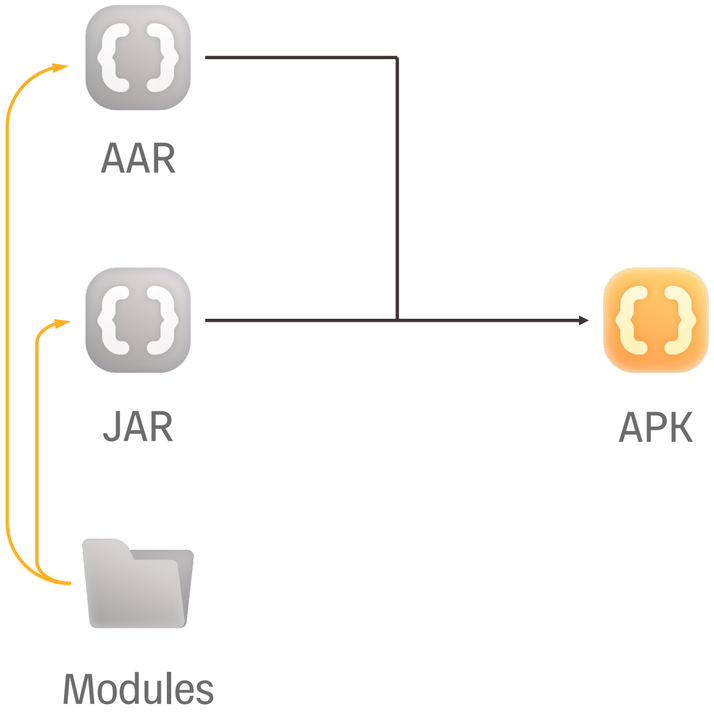
02 Работа с зависимостями в Gradle

03 Билд варианты Android: flavors,...



Сборка APK

- **A**ndroid **P**ackage **K**it (APK)
- **A**ndroid **AR**chive (AAR)
- **J**ava **AR**chive (JAR)
- Multi-module project

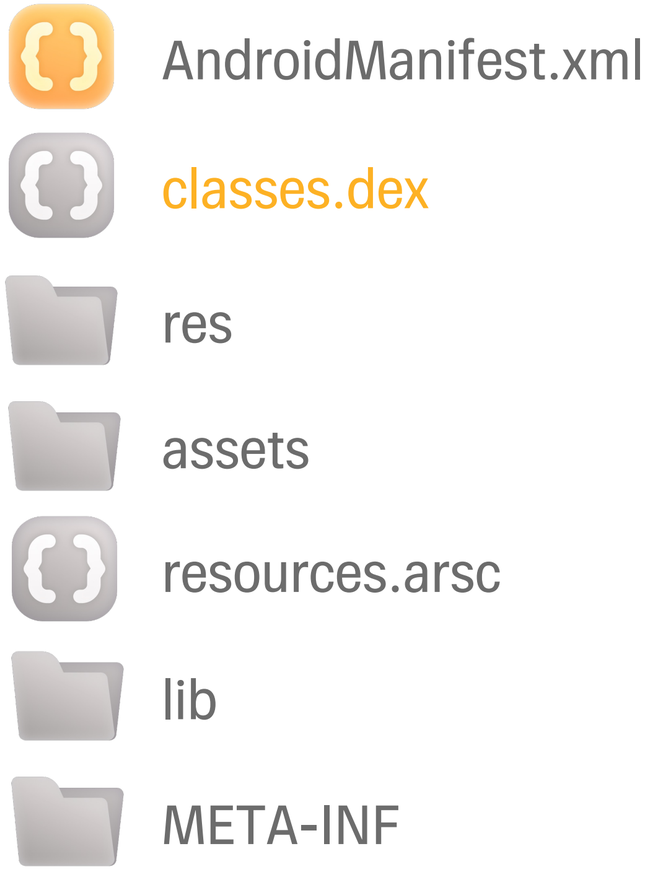


Сборка APK

- **A**ndroid **P**ackage **K**it (APK)
- **A**ndroid **AR**chive (AAR)
- **J**ava **AR**chive (JAR)
- Multi-module project

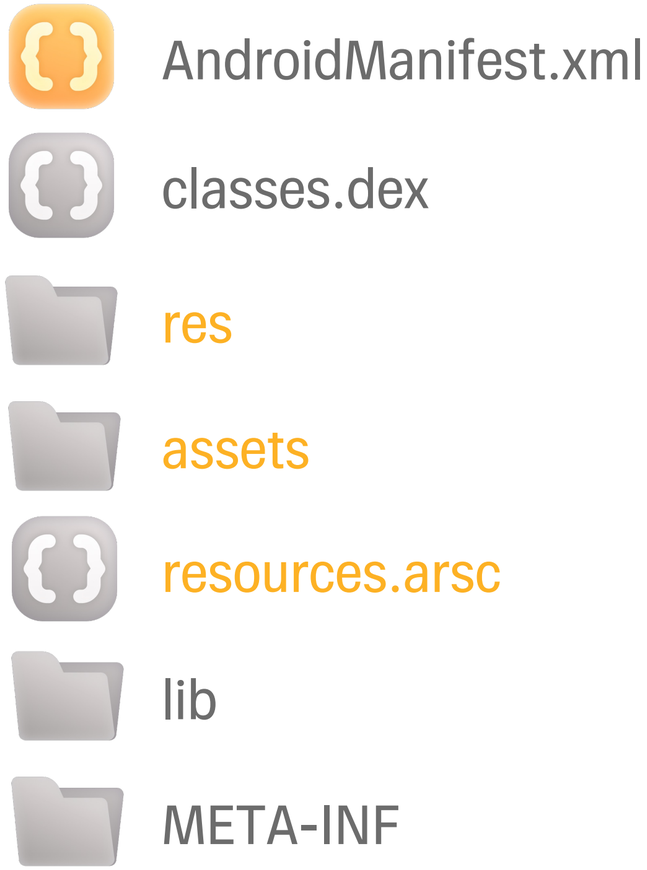
Структура APK

- ZIP архив
- Наличие манифеста
- Все файлы приложения



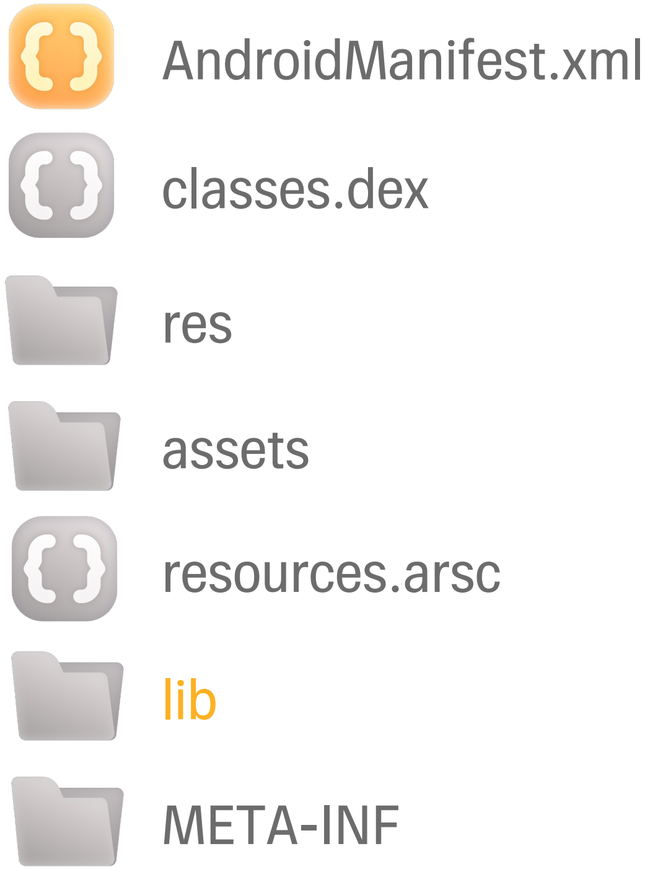
Структура APK

- ZIP архив
- Наличие манифеста
- Все файлы приложения



Структура APK

- ZIP архив
- Наличие манифеста
- Все файлы приложения



Структура APK

- ZIP архив
- Наличие манифеста
- Все файлы приложения

AndroidManifest.xml

classes.dex

res

assets

resources.arsc

lib

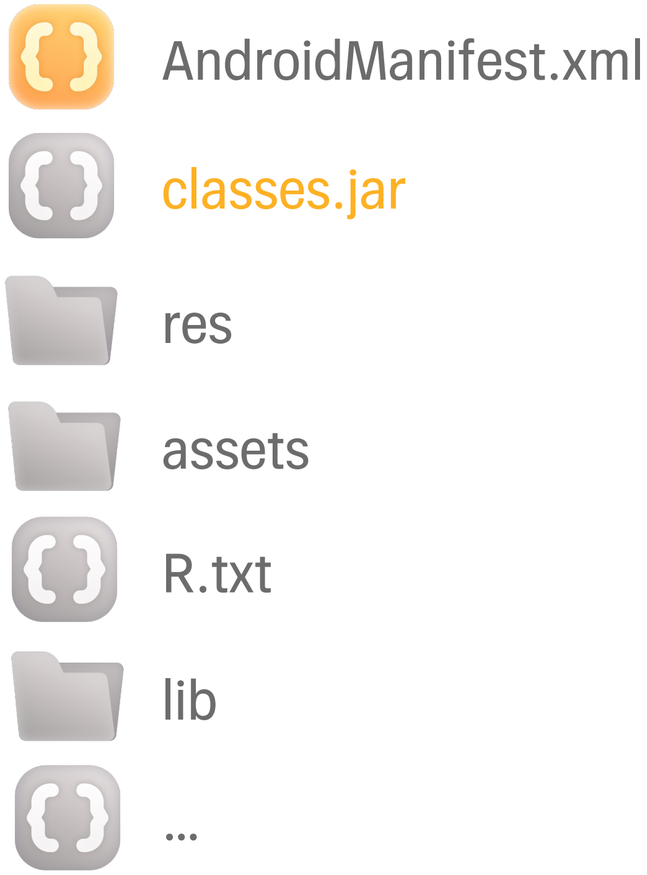
META-INF

Структура APK

- ZIP архив
- Наличие манифеста
- Все файлы приложения

Android ARchive

- ZIP архив
- Наличие манифеста
- Код, ресурсы, библиотеки, ...
- Служебные файлы

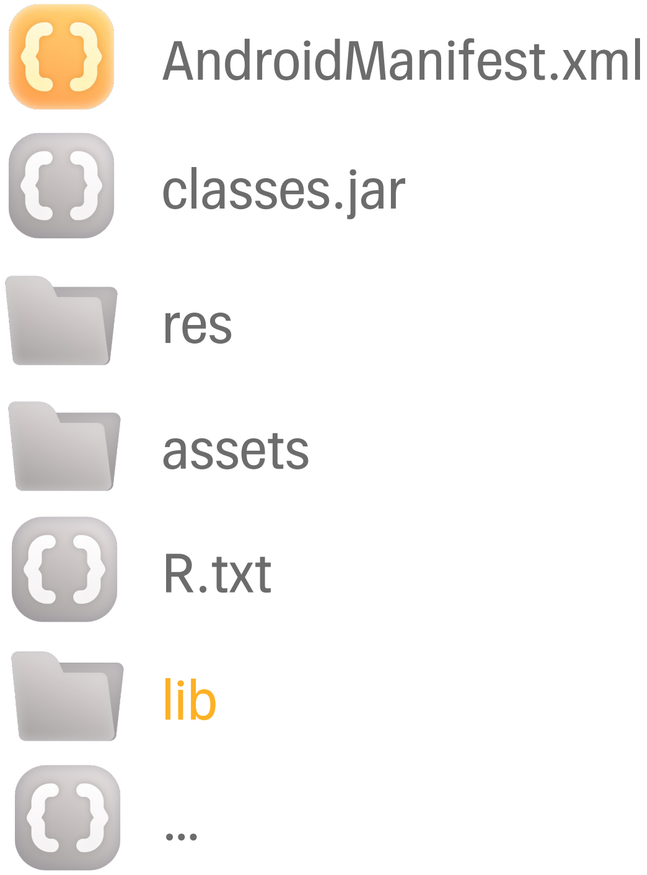


Android ARchive

- ZIP архив
- Наличие манифеста
- Код, ресурсы, библиотеки, ...
- Служебные файлы

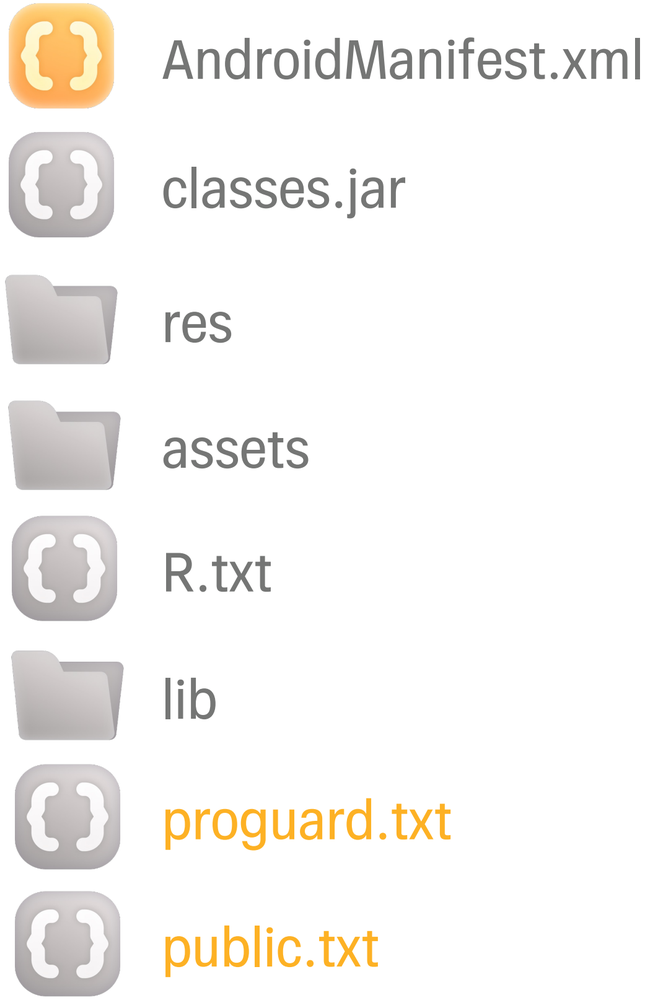
Android ARchive

- ZIP архив
- Наличие манифеста
- Код, ресурсы, библиотеки, ...
- Служебные файлы



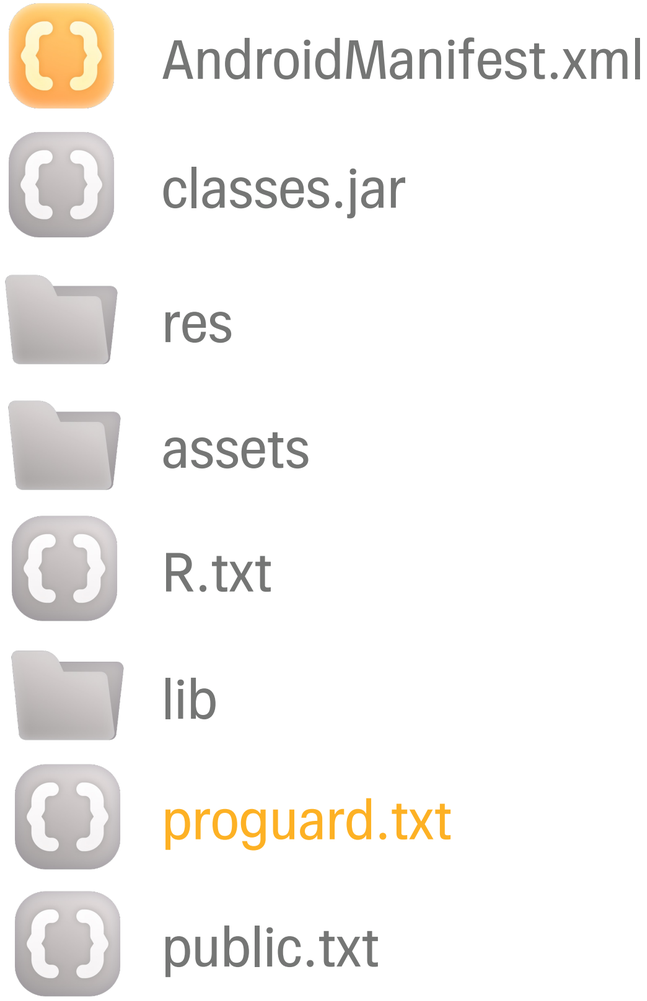
Android ARchive

- ZIP архив
- Наличие манифеста
- Код, ресурсы, библиотеки, ...
- Служебные файлы



Android ARchive

- ZIP архив
- Наличие манифеста
- Код, ресурсы, библиотеки, ...
- Служебные файлы



Android ARchive

- ZIP архив
- Наличие манифеста
- Код, ресурсы, библиотеки, ...
- Служебные файлы

public.txt

- Ограничение области видимости
- Lint правило
- Дополнение кода в IDE

Java ARchive

- ZIP архив
- Может содержать META-INF
- Скомпилированные Java классы

Java ARchive

- ZIP архив
- Может содержать META-INF
- Скомпилированные Java классы

Java ARchive

- ZIP архив
- Может содержать META-INF
- Скомпилированные Java классы

Java ARchive

- ZIP архив
- Может содержать META-INF
- Скомпилированные Java классы

Java ARchive

- ZIP архив
- Может содержать META-INF
- Скомпилированные Java классы

Java ARchive

- ZIP архив
- Может содержать META-INF
- Скомпилированные Java классы

Откуда взялись DLL файлы?

Java ARchive

- ZIP архив
- Может содержать META-INF
- Скомпилированные Java классы
- Ресурсы и прочие файлы

Как поступать с дубликатами артефактов?

01 Файлы AndroidManifest.xml

02 Скомпилированный код

03 Android ресурсы и ассеты

04 Нативные библиотеки (so файлы)

05 Правила обфускации (proguard)

06 Java ресурсы и прочие файлы

Мержинг манифеста

Собрать в один файл

- Обойти все файлы по приоритетам
- Объединить XML элементы

Разрешить конфликты

- Логически сравнить XML элементы
- Объединить атрибуты по приоритетам

Мержинг манифеста

Особые случай разрешения конфликтов

<manifest>

Используется из манифеста
с наивысшим приоритетом

<uses-feature>, <uses-library>

Содержат атрибут required.
Объединяются по принципу ИЛИ

<uses-sdk>

Используется из манифеста
с наивысшим приоритетом

<intent-filter>

Не объединяются между собой.
Всегда считаются уникальными.

Возможные последствия

Неявное объединение атрибутов

Неявное добавление фильтров

Неявное добавление разрешений

Мержинг ресурсов

AGP «молча» переписывает дубликаты

Устранение дубликатов

- Склеивает файлы из values/
- Объединяет res/ и assets/

Приоритет ресурсов

- Наибольший у модуля приложения
- Библиотеки в порядке добавления

Возможные последствия

Неявное использование ресурсов

Библиотека будет использовать какой-то ресурс

Неявное добавление файлов

Раздувание размера APK

Рост потребления памяти

Для AGP версии ниже 9.1

Нативные библиотеки и прочие файлы

Источники данных

- SO файлы из директорий lib/
- Файлы из JAR библиотек

Стратегия объединения

- Собрать все вместе
- Разрешать конфликты вручную

Стратегии объединения для файлов и библиотек

excludes

Исключить добавления указанных файлов в APK

pickFirsts

Из всех файлов оставить только первый попавшийся

merges

Склеить дубликаты в один файл (для Java ресурсов)

Возможные последствия

Неявный выбор библиотек

Отсутствует гарантия бинарной совместимости

Неявное добавление файлов

Раздувание размера APK

Появление лишних файлов

«Мусор» в корне APK файла

Файлы proguard (R8)

Источники данных

- proguard.txt (aar)
- META-INF/proguard/ (jar)

Стратегия объединения

- Собрать все в один файл
- Добавить «встроенные» правила

Удобство разработки

VS

Неявные последствия

Простота интеграции

Библиотека поставляет правила, чтобы избежать дополнительной настройки при подключении

Область действия правил

Правило из одной библиотеки может отключить обфускацию для всего приложения

Скомпилированный код class файлы

Объединить весь код

- classes.jar (aar)
- Java библиотеки

Побочный эффект

Может появиться код,
который не используется в приложении

Возможные источники ошибок

01 Файлы AndroidManifest.xml

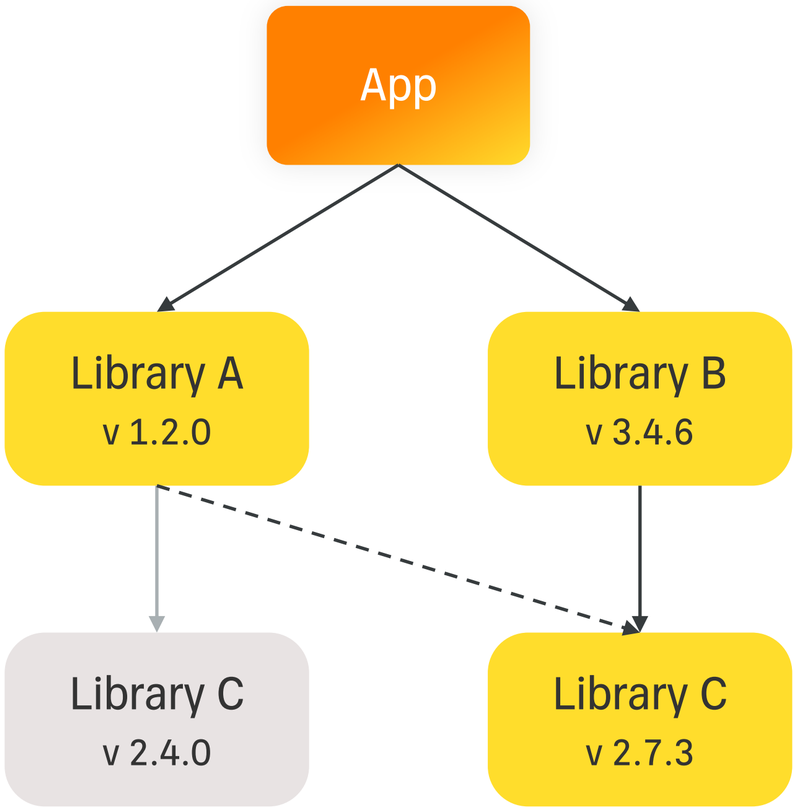
02 Скомпилированный код

03 Android ресурсы и ассеты

04 Нативные библиотеки (so файлы)

05 Правила обфускации (proguard)

06 Java ресурсы и прочие файлы

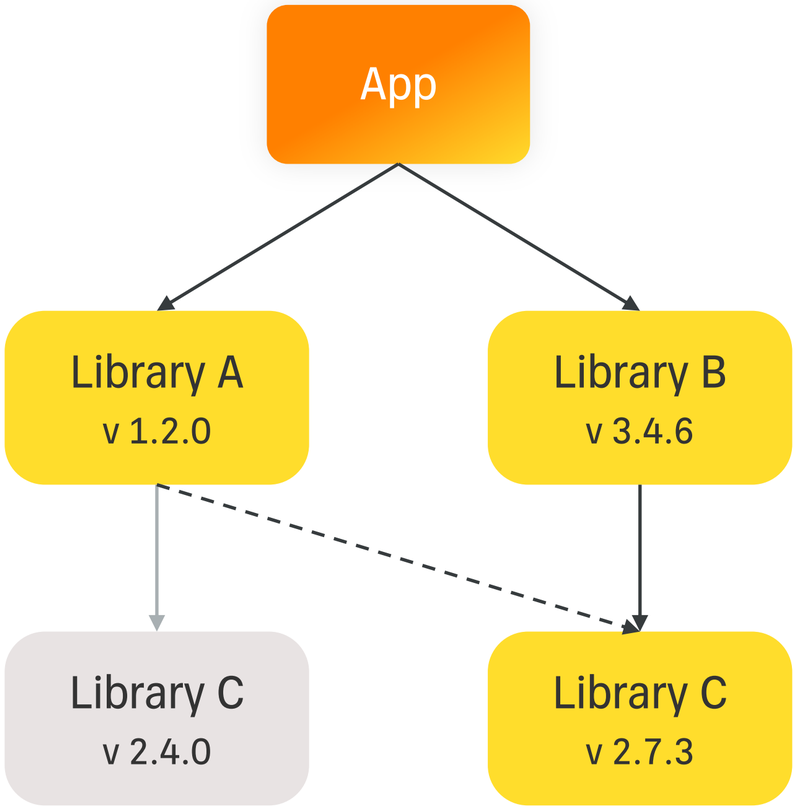


Разрешение зависимостей

Gradle по умолчанию выбирает самую последнюю версию библиотеки.

Итог:

- Ошибки компиляции
- Ошибки в рантайме



Разрешение зависимостей

Gradle по умолчанию выбирает самую последнюю версию библиотеки.

Итог:

- Ошибки компиляции
- Ошибки в рантайме

Никому нельзя доверять!

Возможные последствия

Неявный выбор версии

Gradle самостоятельно решает конфликты

Неявное подключение библиотеки

Транзитивные зависимости

Неявное изменение артефактов

Другой набор библиотек, другие артефакты

Рукотворные причины ошибок

На этапах интеграции и деплоя собираются «разные» приложения

Release vs. Debug

Андроид через документацию и примеры принуждает использовать билд типы

Применение Flavor’ов

Flavor’ы усложняют сборку, увеличивают количество источников кода и ресурсов

Ускорение пайплайнов

Срезание углов и сокращение проверок. Например, запуск compile вместо assemble

Сборка без обфускации

По нашему опыту, сборка с обфускацией длится в 3-4 раза дольше, чем отладочная

Рукотворные причины ошибок

На этапах интеграции и деплоя собираются «разные» приложения

Release vs. Debug

Андроид через документацию и примеры принуждает использовать билд типы

Применение Flavor’ов

Flavor’ы усложняют сборку, увеличивают количество источников кода и ресурсов

Ускорение пайплайнов

Срезание углов и сокращение проверок. Например, запуск compile вместо assemble

Сборка без обфускации

По нашему опыту, сборка с обфускацией длится в 3-4 раза дольше, чем отладочная

Рукотворные причины ошибок

На этапах интеграции и деплоя собираются «разные» приложения

Release vs. Debug

Андроид через документацию и примеры принуждает использовать билд типы

Применение Flavor’ов

Flavor’ы усложняют сборку, увеличивают количество источников кода и ресурсов

Ускорение пайплайнов

Срезание углов и сокращение проверок. Например, запуск compile вместо assemble

Сборка без обфускации

По нашему опыту, сборка с обфускацией длится в 3-4 раза дольше, чем отладочная

Рукотворные причины ошибок

На этапах интеграции и деплоя собираются «разные» приложения

Release vs. Debug

Андроид через документацию и примеры принуждает использовать билд типы

Применение Flavor’ов

Flavor’ы усложняют сборку, увеличивают количество источников кода и ресурсов

Ускорение пайплайнов

Срезание углов и сокращение проверок. Например, запуск compile вместо assemble

Сборка без обфускации

По нашему опыту, сборка с обфускацией длится в 3-4 раза дольше, чем отладочная

Рукотворные причины ошибок

На этапах интеграции и деплоя собираются «разные» приложения

Release vs. Debug

Андроид через документацию и примеры принуждает использовать билд типы

Применение Flavor’ов

Flavor’ы усложняют сборку, увеличивают количество источников кода и ресурсов

Ускорение пайплайнов

Срезание углов и сокращение проверок. Например, запуск compile вместо assemble

Сборка без обфускации

По нашему опыту, сборка с обфускацией длится в 3-4 раза дольше, чем отладочная

Неявные действия

- Добавление разрешений
- Использование ресурсов
- Выбор SO библиотек
- Применение правил R8

VS

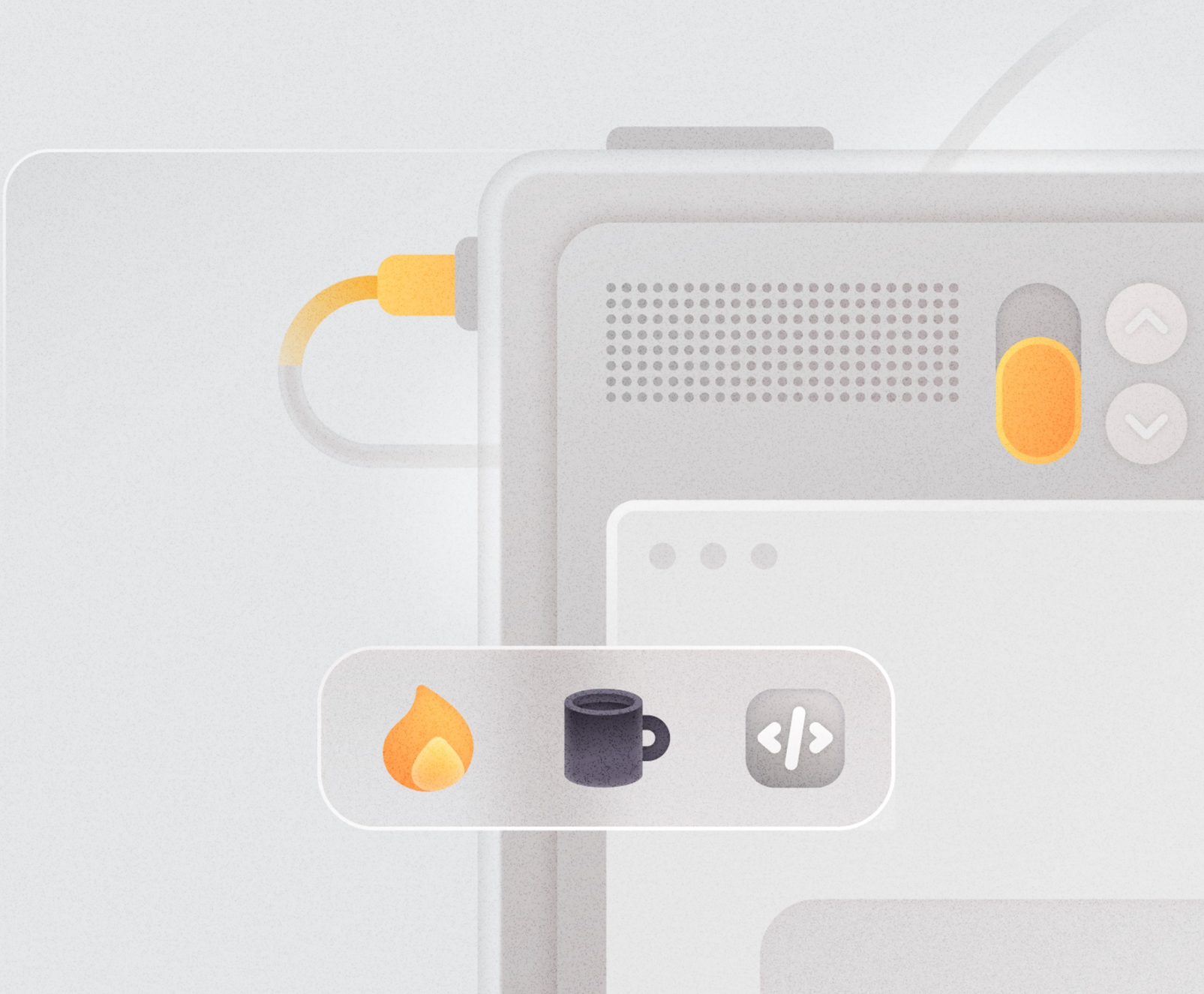
Влияние на разработку

- Отсутствие контроля
- Скрытые взаимосвязи
- Позднее обнаружение

Узнать об ошибке до влития в master

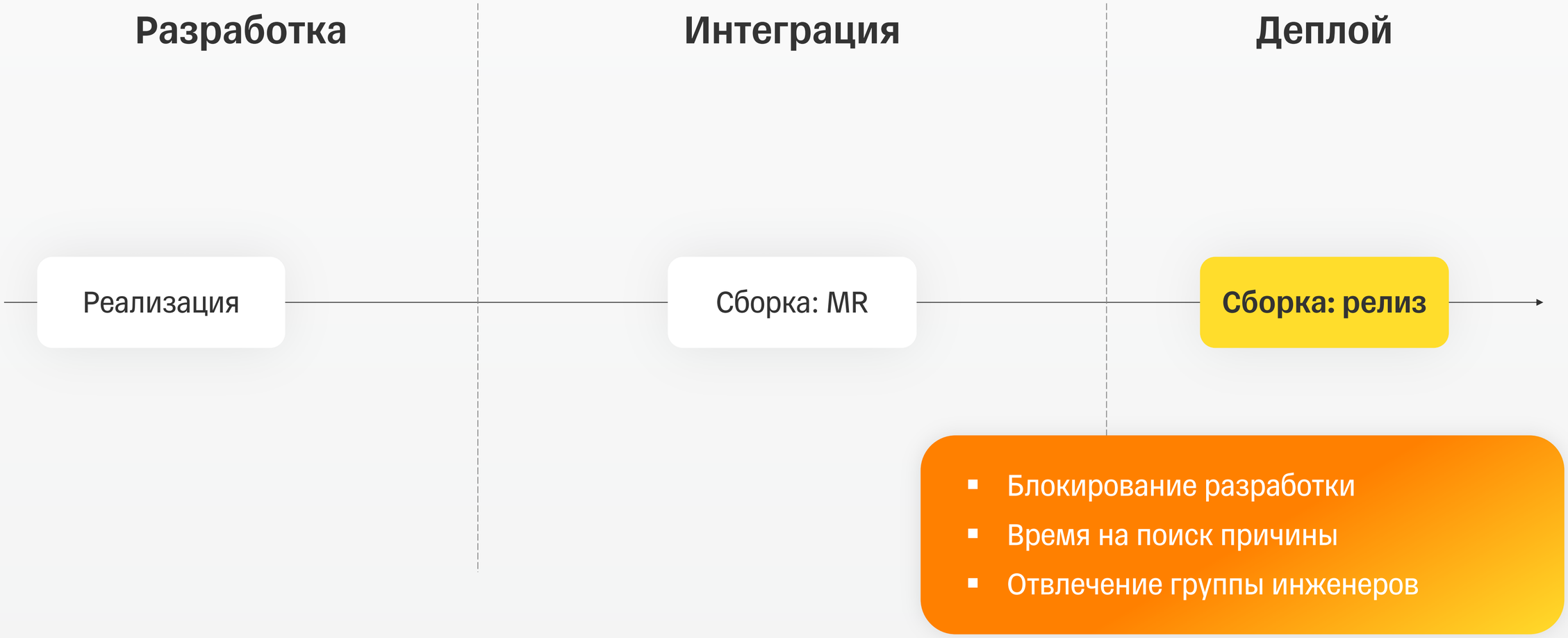
02

Концепция shift-left

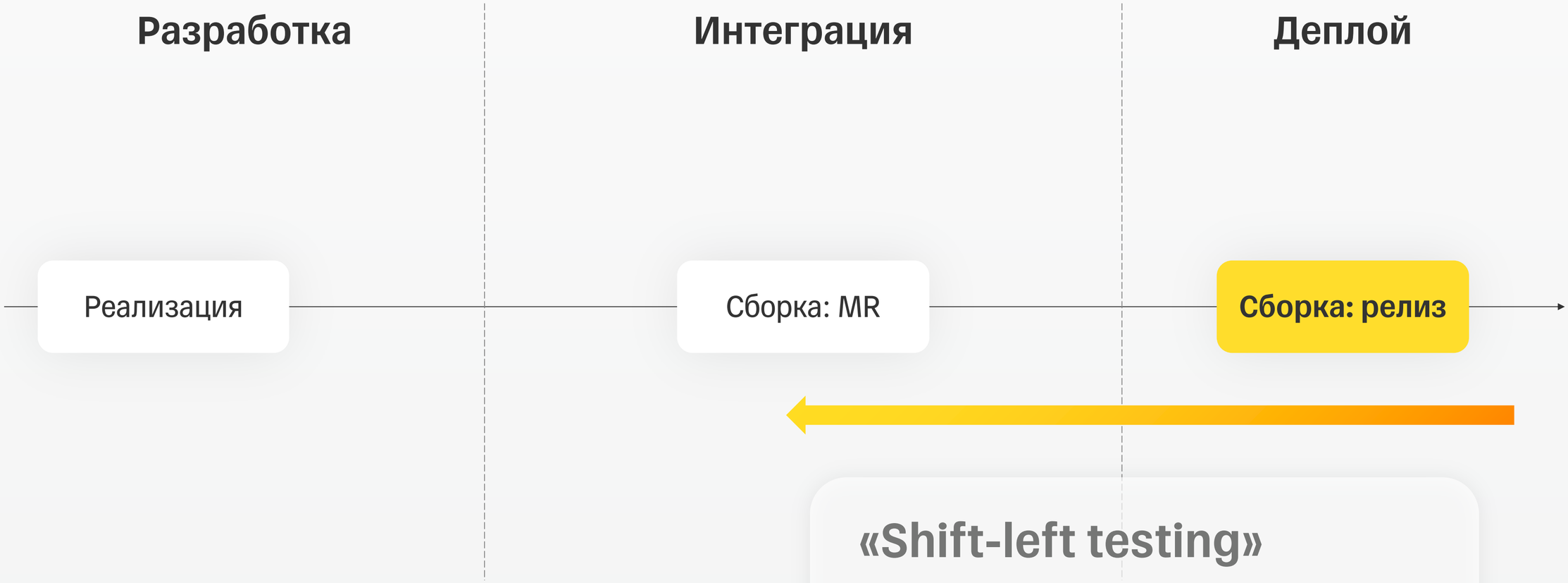


Упрощенный процесс

Упрощенный процесс



Упрощенный процесс

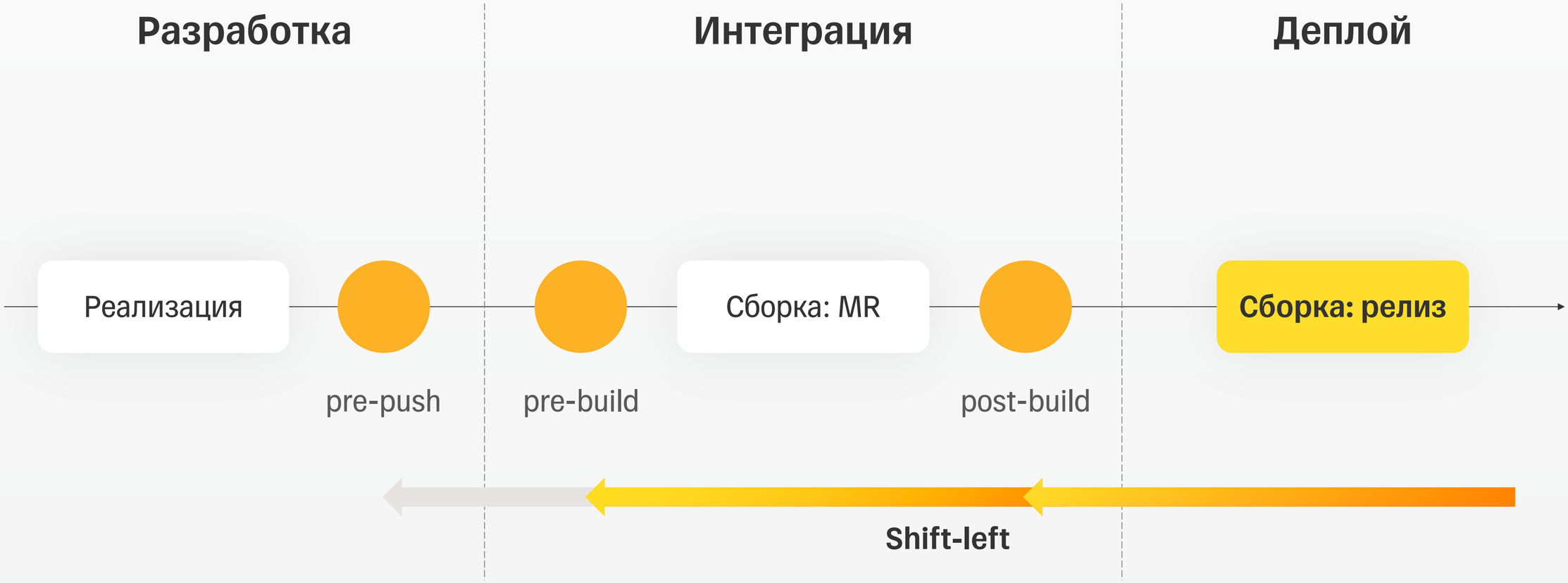


«Shift-left testing»

Ларри Смит (Larry Smith), 2001

Упрощенный процесс

Упрощенный процесс



Выполнение проверок в pre-push

Различия рабочего окружения

Установленные утилиты, версии, ...

Gradle wrapper

Слишком долгая конфигурация билда

Сторонние инструменты

Написать на Golang. Как распространять?

Разработчику

Недостатки

- больше ошибок пайплайна
- отвлечение для исправления

VS

Для команды

Преимущества

- локализация ошибки
- снижение затрат на исправление

Реализация shift-left

Встроить проверки
в пайплайн

01 Достать нужные данные, обработать, сохранить эталонное состояние

02 Получить текущие значения, сверить с эталонным состоянием

03 Устранить причину расхождения или обновить эталон

Основная задача: достать нужную информацию

Из APK файла

Можно достать все нужные данные. Теряется источник исходных файлов

Из репозитория

Самой простой способ по реализации. Отсутствует информация из библиотек

Из зависимостей

Источник всех данных в APK. Доступ через Gradle без сборки приложения

03

Возможности Gradle и AGP

Обход всех подпроектов

- Нарушает изоляцию модулей
- Ломает кэш конфигурации
- Конфигурирует весь проект

build.gradle

```
tasks.register("some_task") {
    subprojects.forEach { subproject ->
        subproject.configurations.forEach { conf ->
            conf.dependencies.forEach { dependency ->
                ...
            }
        }
    }
}
```

Использование конфигураций в Gradle

Сколько конфигураций в проекте «Now in Android»?

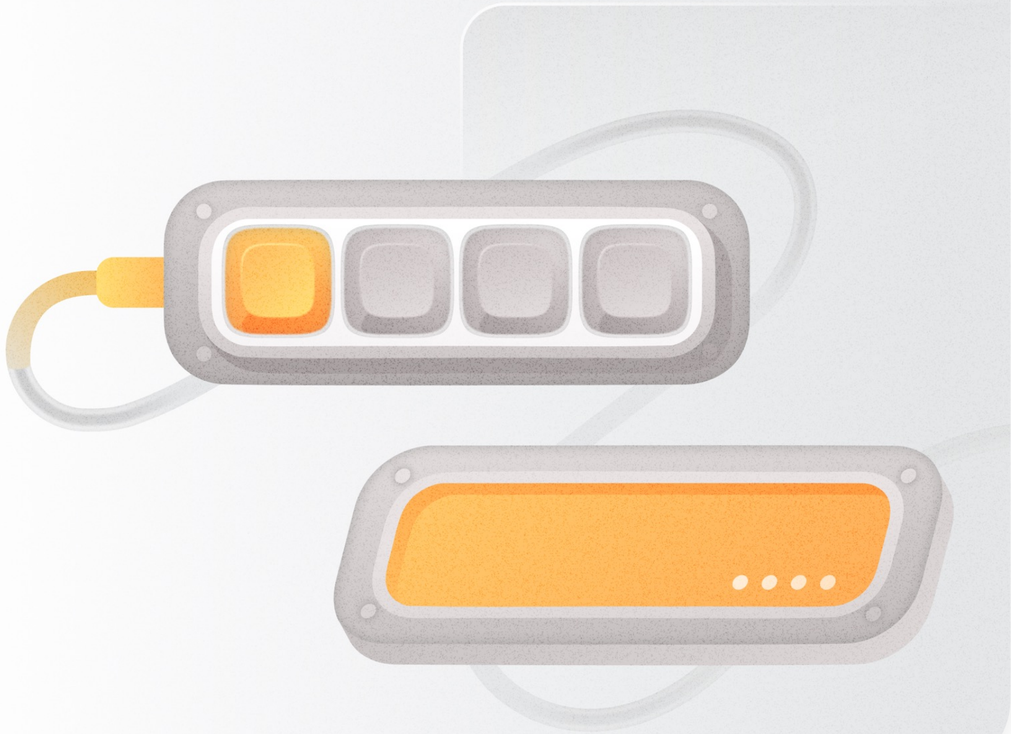
23

122

500+

api, kapt, implementation, ...

Виды конфигураций



Declarable configuration

Объявление зависимостей, ограничение области видимости (`implementation`)

Resolvable configuration

Определяют артефакты необходимые для сборки (`runtimeClasspath`)

Consumable configuration

Предоставляют артефакты для сборки другим модулям (`runtimeElements`)

Конфигурации «Now in Android»

Resolvable

- compileClasspath
- runtimeClasspath
- testCompileClasspath
- testRuntimeClasspath

Consumable

- archives
- default
- apiElements
- mainSourceElements
- runtimeElements
- testResultsElementsForTest

Declarable

- api
- baselineProfile
- debugImplementation
- compileOnly
- composeMappingProducerClasspath
- coreLibraryDesugaring
- hiltAnnotationProcessorProdRelease
- implementation
- kotlin-extension
- ksp
- lintChecks
- testFixturesRuntimeOnly
- ...

Компоненты в Gradle

build.gradle

```
dependencies {
    implementation(project(":utils"))
    implementation("com.squareup.okhttp3:okhttp:5.3.0")
    ...
}
```

Варианты в Gradle

module

```
{
  "component": {
    "group": "com.squareup.okhttp3",
    "module": "okhttp",
    "version": "5.3.0",
  },
  "variants": [
    {"name": "runtimeElements", "files": [{"name": "okhttp-5.3.0.jar", ... }] },
    {"name": "javadocElements", "files": [{"name": "okhttp-5.3.0-javadoc.jar"}] },
    {"name": "sourcesElements", "files": [{"name": "okhttp-5.3.0-sources.jar"}] }
  ]
}
```

Варианты в Gradle

module

```
{
  "component": {
    "group": "com.squareup.okhttp3",
    "module": "okhttp",
    "version": "5.3.0",
  },
  "variants": [
    {"name": "runtimeElements", "files": [{"name": "okhttp-5.3.0.jar", ... }] },
    {"name": "javadocElements", "files": [{"name": "okhttp-5.3.0-javadoc.jar"}] },
    {"name": "sourcesElements", "files": [{"name": "okhttp-5.3.0-sources.jar"}] }
  ]
}
```

Атрибуты

consumer × producer

```
% ./gradlew :app:resolvableConfigurations

--------------------------------------------------
Configuration runtimeClasspath
--------------------------------------------------
Runtime classpath of 'main'.

Attributes
    - org.gradle.category                 = library
    - org.gradle.dependency.bundling      = external
    - org.gradle.jvm.environment          = standard-jvm
    - org.gradle.jvm.version              = 21
    - org.gradle.libraryelements          = jar
    - org.gradle.usage                    = java-runtime
    - org.jetbrains.kotlin.platform.type = jvm
Extended Configurations
    - implementation
    - runtimeOnly
```

consumer producer ×

```
% ./gradlew :utils:outgoingVariants

--------------------------------------------------
Variant runtimeElements
--------------------------------------------------
Runtime elements for the 'main' feature.

Capabilities
    - utils:unspecified (default capability)
Attributes
    - org.gradle.category                 = library
    - org.gradle.dependency.bundling      = external
    - org.gradle.jvm.environment          = standard-jvm
    - org.gradle.jvm.version              = 21
    - org.gradle.libraryelements          = jar
    - org.gradle.usage                    = java-runtime
    - org.jetbrains.kotlin.platform.type = jvm
Artifacts
    - build/libs/utils.jar (artifactType = jar)
```

Атрибуты

consumer × producer

```
% ./gradlew :app:resolvableConfigurations

--------------------------------------------------
Configuration runtimeClasspath
--------------------------------------------------
Runtime classpath of 'main'.

Attributes
    - org.gradle.category                = library
    - org.gradle.dependency.bundling     = external
    - org.gradle.jvm.environment         = standard-jvm
    - org.gradle.jvm.version             = 21
    - org.gradle.libraryelements         = jar
    - org.gradle.usage                   = java-runtime
    - org.jetbrains.kotlin.platform.type = jvm
Extended Configurations
    - implementation
    - runtimeOnly
```

consumer producer ×

```
% ./gradlew :utils:outgoingVariants

--------------------------------------------------
Variant runtimeElements
--------------------------------------------------
Runtime elements for the 'main' feature.

Capabilities
    - utils:unspecified (default capability)
Attributes
    - org.gradle.category                = library
    - org.gradle.dependency.bundling     = external
    - org.gradle.jvm.environment         = standard-jvm
    - org.gradle.jvm.version             = 21
    - org.gradle.libraryelements         = jar
    - org.gradle.usage                   = java-runtime
    - org.jetbrains.kotlin.platform.type = jvm
Artifacts
    - build/libs/utils.jar (artifactType = jar)
```

Артефакты

consumer × producer

```
% ./gradlew :app:resolvableConfigurations

--------------------------------------------------
Configuration runtimeClasspath
--------------------------------------------------
Runtime classpath of 'main'.

Attributes
    - org.gradle.category                 = library
    - org.gradle.dependency.bundling      = external
    - org.gradle.jvm.environment          = standard-jvm
    - org.gradle.jvm.version              = 21
    - org.gradle.libraryelements          = jar
    - org.gradle.usage                    = java-runtime
    - org.jetbrains.kotlin.platform.type = jvm
Extended Configurations
    - implementation
    - runtimeOnly
```

consumer producer ×

```
% ./gradlew :utils:outgoingVariants

--------------------------------------------------
Variant runtimeElements
--------------------------------------------------
Runtime elements for the 'main' feature.

Capabilities
    - utils:unspecified (default capability)
Attributes
    - org.gradle.category                 = library
    - org.gradle.dependency.bundling      = external
    - org.gradle.jvm.environment          = standard-jvm
    - org.gradle.jvm.version              = 21
    - org.gradle.libraryelements          = jar
    - org.gradle.usage                    = java-runtime
    - org.jetbrains.kotlin.platform.type = jvm
Artifacts
    - build/libs/utils.jar (artifactType = jar)
```

Можно получить артефакты без обхода модулей

Доступ к Jar файлам
ArtifactView API

build.gradle ×

```
tasks.register<ResolveFiles>("resolveSources") {
    files.from(configurations.runtimeClasspath.map {
        it.incoming.artifactView {
            withVariantReselection()
            componentFilter { ... }

            attributes {
                attribute(ArtifactTypeDefinition.ARTIFACT_TYPE_ATTRIBUTE, "jar")
            }
        }.files
    })
}
```

Доступ к Jar файлам
ArtifactView API

build.gradle

```
tasks.register<ResolveFiles>("resolveSources") {
    files.from(configurations.runtimeClasspath.map {
        it.incoming.artifactView {
            withVariantReselection()
            componentFilter { ... }

            attributes {
                attribute(ArtifactTypeDefinition.ARTIFACT_TYPE_ATTRIBUTE, "jar")
            }
        }.files
    })
}
```

Возможности ArtifactView API

build.gradle

```
tasks.register<ResolveFiles>("resolveSources") {
    files.from(configurations.runtimeClasspath.map {
        it.incoming.artifactView {
            withVariantReselection()
            componentFilter { ... }

            attributes {
                attribute(ArtifactTypeDefinition.ARTIFACT_TYPE_ATTRIBUTE, "jar")
            }
        }.files
    })
}
```

Возможности ArtifactView API

build.gradle

```
tasks.register<ResolveFiles>("resolveSources") {
    files.from(configurations.runtimeClasspath.map {
        it.incoming.artifactView {
            withVariantReselection()
            componentFilter { ... }

            attributes {
                attribute(ArtifactTypeDefinition.ARTIFACT_TYPE_ATTRIBUTE, "jar")
            }
        }.files
    })
}
```

Подготовленные артефакты Secondary variants

producer

```
------------------------------------------------------------
Secondary Variant android-assets
------------------------------------------------------------
    Attributes
        - com.android.build.api.attributes.AgpVersionAttr          = 8.13.2
        - com.android.build.api.attributes.BuildTypeAttr           = release
        - com.android.build.gradle.internal.attributes.VariantAttr = release
        - org.gradle.category                                      = library
        - org.gradle.jvm.environment                               = android
        - org.gradle.usage                                         = java-runtime
        - org.jetbrains.kotlin.platform.type                       = androidJvm
    Artifacts
        - build/intermediates/assets/release/mergeReleaseAssets (artifactType = android-assets)
```

Доступ к подготовленным артефактам

build.gradle

```
tasks.register<ResolveFiles>("resolveSources") {
    files.from(configurations.runtimeClasspath.map {
        it.incoming.artifactView {
            withVariantReselection()
            componentFilter { ... }

            attributes {
                attribute(AndroidArtifacts.ARTIFACT_TYPE, ArtifactType.ASSETS.type)
            }
        }.files
    })
}
```

Android Gradle Plugin sources

Artifacts

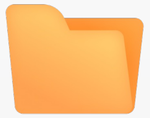
```
public class AndroidArtifacts {
    // types published by an Android library
    private static final String TYPE_CLASSES_JAR = "android-classes-jar"; // In AAR
    private static final String TYPE_CLASSES_DIR = "android-classes-directory"; // Not in AAR
    private static final String TYPE_SHARED_CLASSES = "android-shared-classes";
    private static final String TYPE_DEX = "android-dex";
    private static final String TYPE_D8_OUTPUTS = "android-d8-outputs";
    private static final String TYPE_KEEP_RULES = "android-keep-rules";
    private static final String TYPE_MANIFEST = "android-manifest";
    private static final String TYPE_MANIFEST_METADATA = "android-manifest-metadata";
    private static final String TYPE_ANDROID_RES = "android-res";
    ...
}
```

AGP: Variant API

https://github.com/android/gradle-recipes

Настройка билда

Управление настройками билда, которые обычно задаются в DSL



Доступ к артефактам

Доступ к class файлам, манифесту, ..., финальным артефактам (APK, AAB)

Variant API

- Android Gradle Plugin API
- Артефакты в текущем модуле

VS

ArtifactView API

- Gradle API
- Данные об источнике файлов

04

Как неявное сделать явным

Готовые решения для отдельных задач

dependency-guard

Контроль изменения зависимостей

В документации Android

manifest-guard

Обнаружение изменений в манифесте

Решение от Авито

Изменения правил Proguard

Включение
Global options

01 `-allowaccessmodification`

02 `-dontobfuscate`

03 `-dontoptimize`

04 `-dontshrink`

05 `-repackageclasses`

06 `-ignorewarnings`

Отключение логирования в Android

proguard

```
-assumenosideeffects class android.util.Log {
    public static *** d(...);
    public static *** v(...);
    public static *** i(...);
    public static *** w(...);
    public static *** e(...);
    public static *** wtf(...);
    public static *** println(...);
}
```

Использование «широких» правил

proguard

```
-if @kotlinx.serialization.Serializable class **
-keep, allowshrinking, allowoptimization, allowobfuscation, allowaccessmodification class <1>
```

Использование «широких» правил

proguard

```
-if @kotlinx.serialization.Serializable class **
-keep, allowshrinking, allowoptimization, allowobfuscation, allowaccessmodification class <1>

-keep @kotlinx.serialization.Serializable class * { *; }
```

+500k ссылок

+48%, 1080k → 1600k

+14 Мб к размеру

+6%, 223 Мб → 237 Мб

Функции R8

01 Минификация

02 Обфускация

03 Оптимизация

Применение правил к пакету

proguard

```
-keep class com.yandex.mapkit.** { *; }
-keep interface com.yandex.mapkit.** { *; }
```

Применение правил к пакету

proguard

```
-keep class com.yandex.mapkit.** { *; }
-keep interface com.yandex.mapkit.** { *; }


-keep,allowobfuscation class com.yandex.mapkit.map.CameraListener { *; }
-keep,allowobfuscation class * extends com.yandex.mapkit.map.CameraListener { *; }
-keepclassmembers class * {
    com.yandex.mapkit.map.CameraListener *;
}
```

R8 оптимизирует код

before × after

```
class MapEngine( ... ) : MapView {

    val listener = CameraListener { ... }

    override fun renderMap() {
        ...
        map.addCameraListener(listener)
        ...
    }
}
```


before after ×

```
class MapEngine( ... ) : MapView {

    override fun renderMap() {
        ...
        map.addCameraListener(
            CameraListener { ... }
        )
        ...
    }
}
```

R8 оптимизирует код

before × after

```
class MapEngine( ... ) : MapView {

    val listener = CameraListener { ... }

    override fun renderMap() {
        ...
        map.addCameraListener(listener)
        ...
    }
}
```

before after ×

```
class MapEngine( ... ) : MapView {

    override fun renderMap() {
        ...
        map.addCameraListener(
            CameraListener { ... }
        )
        ...
    }
}
```

R8 оптимизирует код

before × after

```
class MapEngine( ... ) : MapView {

    val listener = CameraListener { ... }

    override fun renderMap() {
        ...
        map.addCameraListener(listener)
        ...
    }
}
```

before after ×

```
class MapEngine( ... ) : MapView {

    override fun renderMap() {
        ...
        map.addCameraListener(
            CameraListener { ... }
        )
        ...
    }
}
```

Правильнее – отрефакторить код

Стараться избегать добавления keep правил

Основные причины

- Использование рефлексии
- Вызовы нативного кода

Добавляйте обдуманно

Начинайте с поиска причины в коде.

Хоть это и сильно «больнее» и дольше.

Нововведения Android Gradle Plugin 9.0

proguardAndroidTxt.disallowed

Предотвратить добавления dontoptimize

Использовать proguard-android-optimize.txt

globalOptionsInConsumerRules.disallowed

Игнорировать глобальные опции в библиотеках

Неявно отключить dontobfuscate, dontoptimize, ...

Узнать до влития кода в master

01

Получить текущий набор правил

В идеале все, что R8 использует для обфускации: содержимое configuration.txt

02

Сравнить с прежним состоянием

Хранить в репозитории, чтобы можно было легко сверить изменения

03

Определить место добавления правил

При необходимости понять: откуда взялось проблемное правило

Источники правил Proguard

Стандартные правила

Базовый набор правил,
proguard-android-optimize.txt.
Генерирует утилита aapt2

Файлы в репозитории

Файлы с правилами,
как в модуле приложения,
так и библиотечных модулях

Библиотеки aar и jar

Файлы proguard.txt
и содержимое директорий
META-INF/proguard

Аннотации Keep

Классы, методы, ...
помеченные специальной
аннотацией @Keep

Источники правил Proguard

Стандартные правила

Базовый набор правил,
proguard-android-optimize.txt.
Генерирует утилита aapt2

Файлы в репозитории

Файлы с правилами,
как в модуле приложения,
так и библиотечных модулях

Библиотеки aar и jar

Файлы proguard.txt
и содержимое директорий
META-INF/proguard

Аннотации Keep

Классы, методы, ...
помеченные специальной
аннотацией @Keep

Источники правил Proguard

Стандартные правила

Базовый набор правил,
proguard-android-optimize.txt.
Генерирует утилита aapt2

Файлы в репозитории

Файлы с правилами,
как в модуле приложения,
так и библиотечных модулях

Библиотеки aar и jar

Файлы proguard.txt
и содержимое директорий
META-INF/proguard

Аннотации Keep

Классы, методы, ...
помеченные специальной
аннотацией @Keep

Источники правил Proguard

Стандартные правила

Базовый набор правил,
proguard-android-optimize.txt.
Генерирует утилита aapt2

Файлы в репозитории

Файлы с правилами,
как в модуле приложения,
так и библиотечных модулях

Библиотеки aar и jar

Файлы proguard.txt
и содержимое директорий
META-INF/proguard

Аннотации Keep

Классы, методы, ...
помеченные специальной
аннотацией @Keep

Gradle таска с использованием Configuration avoidance API

build.gradle

```
tasks.register(
    name = ”<task_name>",
    type = CheckProguardRulesTask::class.java,
    configurationAction = CheckProguardTaskAction(<resolvable_configuration>, ...)
)
```

Gradle таска с использованием Configuration avoidance API

build.gradle

```
tasks.register(
    name = "<task_name>",
    type = CheckProguardRulesTask::class.java,
    configurationAction = CheckProguardTaskAction(<resolvable_configuration>, ...)
)
```

Gradle таска с использованием Configuration avoidance API

build.gradle

```
tasks.register(
    name = ”<task_name>",
    type = CheckProguardRulesTask::class.java,
    configurationAction = CheckProguardTaskAction(<resolvable_configuration>, ...)
)
```

Build cache
Lazy Properties

task.kt

```
@CacheableTask
abstract class CheckProguardRulesTask : DefaultTask() {

    @get:Input
    abstract val artifacts: MapProperty<String, String> // <artifact, file>

    @get:PathSensitive(PathSensitivity.RELATIVE)
    @get:InputDirectory
    abstract val baselineDir: DirectoryProperty

    @TaskAction
    fun checkProguardRules() { ... }
}
```

Build cache
Lazy Properties

task.kt

```
@CacheableTask
abstract class CheckProguardRulesTask : DefaultTask() {

    @get:Input
    abstract val artifacts: MapProperty<String, String> // <artifact, file>

    @get:PathSensitive(PathSensitivity.RELATIVE)
    @get:InputDirectory
    abstract val baselineDir: DirectoryProperty

    @TaskAction
    fun checkProguardRules() { ... }
}
```

Build cache
Lazy Properties

task.kt

```
@CacheableTask
abstract class CheckProguardRulesTask : DefaultTask() {

    @get:Input
    abstract val artifacts: MapProperty<String, String> // <artifact, file>

    @get:PathSensitive(PathSensitivity.RELATIVE)
    @get:InputDirectory
    abstract val baselineDir: DirectoryProperty

    @TaskAction
    fun checkProguardRules() { ... }
}
```

Вычисление параметров и передача в таску

action.kt

```
internal class CheckProguardTaskAction(
    targetConfiguration: Configuration, // runtimeClasspath
    ...
) : Action<CheckProguardRulesTask> {

    override fun execute(t: CheckProguardRulesTask) {
        t.artifacts.set(targetConfiguration.proguardRules())
        t.baselineRulesDir.set(...)
        ...
    }
}
```

Получение Proguard правил из всех зависимостей

action.kt

```
internal fun Configuration.proguardRules(): Map<String, String> = this.flatMap { cfg ->
    cfg.incoming.artifactView {
        attributes {
            attribute(
                Attribute.of(AndroidArtifacts.ARTIFACT_TYPE, String::class.java),
                ArtifactType.FILTERED_PROGUARD_RULES
            )
        }
    }.artifacts.resolvedArtifacts.map { artifactResults ->
        val artifactName = ... // название модуля или идентификатор библиотеки
        artifactName to artifactResult.file
    }
}
```

Получение Proguard правил из всех зависимостей

action.kt

```
internal fun Configuration.proguardRules(): Map<String, String> = this.flatMap { cfg ->
    cfg.incoming.artifactView {
        attributes {
            attribute(
                Attribute.of(AndroidArtifacts.ARTIFACT_TYPE, String::class.java),
                ArtifactType.FILTERED_PROGUARD_RULES
            )
        }
    }.artifacts.resolvedArtifacts.map { artifactResults ->
        val artifactName = ... // название модуля или идентификатор библиотеки
        artifactName to artifactResult.file
    }
}
```

Устойчивость baseline и поиск источника

baseline

```
# From ######################## androidx.annotation:annotation-jvm ########################
-keep,allowobfuscation @interface androidx.annotation.Keep
-keep @androidx.annotation.Keep class * {*;}
...
# End of ######################## androidx.annotation:annotation-jvm ########################


# From ######################## androidx.appcompat:appcompat ########################
...
# End of ######################## androidx.appcompat:appcompat ########################
```

Исключение правил из библиотек

build.gradle ×

```
release {
    optimization.keepRules {
        // Ignore all consumer rules
        it.ignoreFrom("com.somelibrary:somelibrary")
    }
}
```

Исключить из baseline и отслеживать изменения

Убрать «белый шум»

Разделить бейзлайны на два:
основной набор правил и исключения

Вернуть после исправления

Переносить изменения в правилах,
отслеживать момент исправления

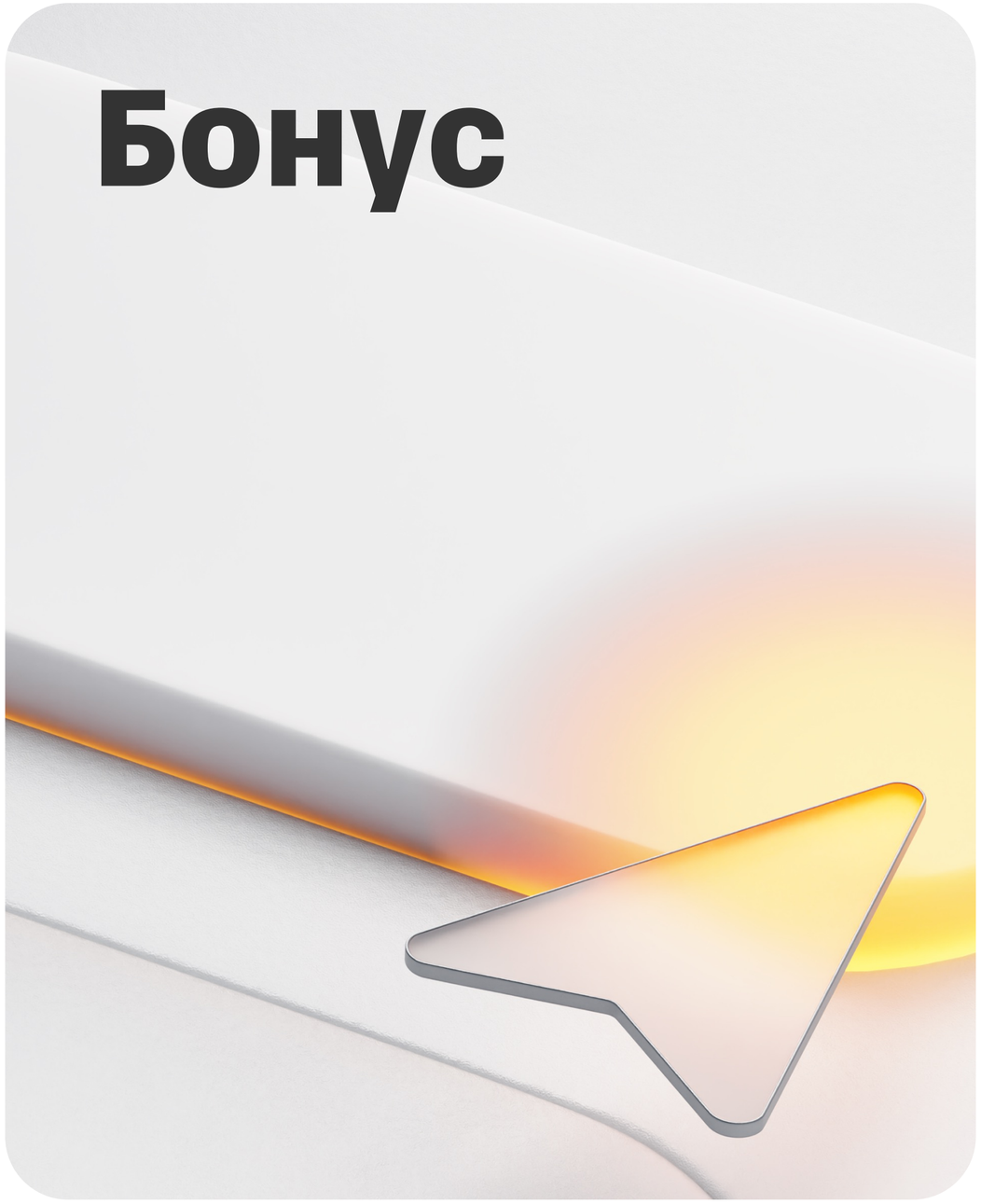
Open Source решение

https://opensource.tbank.ru/mobile-tech/dependency-ripper-plugin

Мы публикуем в Maven Central артефакты, которые используем сами.

Планируем развивать, добавлять проверки на ресурсы
и нативные библиотеки, ...

Бонус



Поиск «мертвого» кода

Отчет по данным из R8 с разбивкой
по модулям, библиотекам, коду и ресурсам

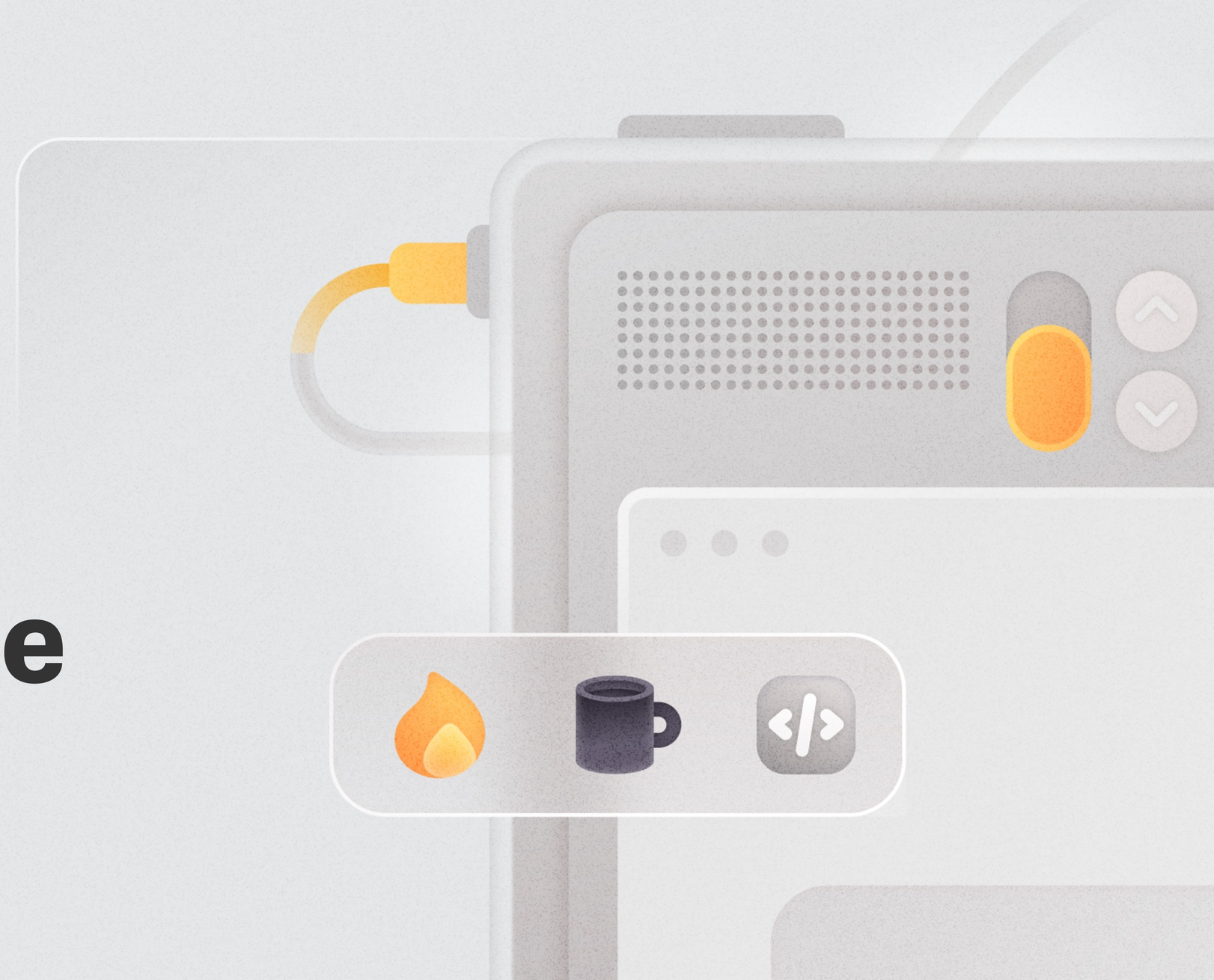
Точки подключения библиотек

Отчет по всем библиотекам. Использование
в модулях, напрямую или транзитивно

Проверка дубликатов классов

Есть случаи, когда встроенная проверка
AGP пропускает ошибки

Подведение итогов



Нюансы сборки Андроид приложения

01 Неявное добавление разрешений

02 Неявное использование ресурсов

03 Неявный выбор so библиотек

04 Неявное добавление файлов

05 Неявное применение правил R8

06 Неявное подключение библиотек

Влияние на процесс разработки

01 **Причина ошибок на поздних этапах разработки**

02 Ошибки не являются следствием намеренных изменений в коде

03 Появление ошибок в фичах никак не связанных с изменениями

Используйте Gradle правильно

01

Забудьте про обход модулей в проекте

Своими руками создаете причины замедления сборки

02

Следуйте гайдам и рекомендациям

Gradle много внимания уделяет перфомансу сборок

03

Используйте API для поиска артефактов

- ArtifactView API
- AGP Variant API

За

Неявное становится явным

Обнаружить проблему раньше,
чем код попадет в master ветку.
Ограничить область влияния ошибки

Против

Дополнительная нагрузка

Дополнительные проверки ведут
к усложнению процессов
и замедляют влитие изменений

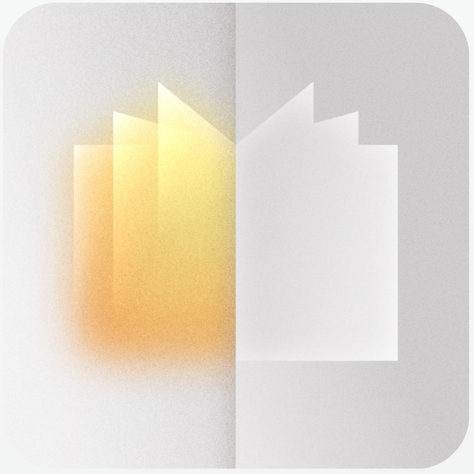
Когда ошибку не видно в мерж реквесте

Бороться системно

Если довелось встречать подобные ошибки

Знать «куда копать»

Когда «посчастливится» решать подобный кейс



Предупрежден – значит вооружен!

No-code
Python
JavaScript
Node.js
Swift
Mobile App
Development
Innovation
Golang
Java
Analysis
Planning

Спасибо за внимание!

Ваши вопросы