ISSN 2072-0297

# МОЛОДОЙ УЧЁНЫЙ

МЕЖДУНАРОДНЫЙ НАУЧНЫЙ ЖУРНАЛ

30
2026
ЧАСТЬ I

16+

# Молодой ученый

Международный научный журнал

№ 30 (633) / 2026

Издается с декабря 2008 г. Выходит еженедельно

На обложке изображен *Владимир Клавдиевич Арсеньев* (1872–1930), русский путешественник, географ, этнограф, писатель, исследователь Дальнего Востока.

Владимир Арсеньев родился в Петербурге. Его отец Клавдий Арсеньев был выходцем из тверских крепостных, дослужился до заведующего движением Московской окружной железной дороги, был удостоен почетного звания потомственного почетного гражданина Санкт-Петербурга.

По примеру своих вдохновителей — путешественников-офицеров Пржевальского и Невельского — Арсеньев избрал военную карьеру. В 1891 году он поступил в 145-й Новочеркасский пехотный полк, а два года спустя — в Петербургское пехотное юнкерское училище. В 1896 году Владимира Арсеньева уже в чине подпоручика направили служить в 14-й Олонецкий пехотный полк, расквартированный в польском городе Ломже.

В 1900 году Арсеньев подал прошение о переводе на почти не изученный в то время Дальний Восток, и его перевели в 1-й Владивостокский крепостной пехотный полк. В следующие годы офицер-путешественник ходил в короткие экспедиции по Дальнему Востоку, в 1900–1905 годах обследовал весь юг Приморья. Уже тогда Арсеньев не только наносил на карту то, что полагалось по заданию военного начальства, но и описывал флору и фауну, археологические находки и этнографические особенности местности.

Несмотря на то что Владимир Арсеньев никогда не учился в университете и закончил только двухгодичное юнкерское училище, с помощью самообразования ему удалось стать компетентным и разносторонним исследователем. Сфера его научных интересов была разнообразна и обширна: он занимался географией и этнографией, картографией, статистикой, археологией, геологией, гидрологией и метеорологией, музейным делом и даже орнитологией и лингвистикой

Во время Русско-японской войны 1904–1905 годов Арсеньев участвовал в разведывательных операциях, был награжден орденами Святой Анны III и IV степени и Святого Станислава III и II степени.

После поражения в войне было решено осваивать территорию Дальнего Востока. В 1906 году приамурский генерал-губернатор Павел Унтербергер выделил средства на первую большую экспедицию — в нее отправили штабс-капитана Владимира Арсеньева, которого к тому моменту перевели в Хабаровск. Во время экспедиции команда Арсеньева изучала горную область Сихотэ-Алиня от залива Святой Ольги до бухты Терней и систему реки Уссури.

Летом 1906 года группа встретила в тайге нанайского охотника Дерсу Узала, который затем стал проводником и другом Арсеньева, а позже — и героем его книг.

В следующей экспедиции 1907 года Владимир Арсеньев продолжил изучать восточные склоны Сихотэ-Алиня и бассейны рек Иман (сегодня — Большая Уссурка) и Бикин. За семь месяцев группа прошла более тысячи верст, пережила две голодовки и зимнюю стужу без теплой одежды, которую унесло с лодкой. Но самой тяжелой стала третья экспедиция 1908–1910 годов: за 19 месяцев путешественники обследовали север Уссурийского края в низовье Амура.

Во время экспедиций Арсеньев привел в единую систему географические наименования Дальнего Востока, указал, как одна и та же река или гора называется у разных народов и какое русское наименование следует использовать. Арсеньев описал быт и верования коренных народов Приамурья — удэгейцев, тазов, орочей, нанайцев.

В 1910–1919 годах Владимир Арсеньев параллельно с военной службой работал директором Хабаровского краеведческого музея.

После переезда во Владивосток в начале 1920-х годов Арсеньев заведовал этнографическим отделом Приморского музея, который сегодня носит его имя. Обширный материал, собранный им во время экспедиций, пополнил коллекции не только Хабаровска и Владивостока. Многие предметы Арсеньев отправлял в дар антропологическому музею Московского университета, этнографическому музею Казанского университета, этнографическому отделу Русского музея. За материалы, отправленные в Вашингтонский музей, путешественника избрали членом Вашингтонского национального географического общества.

В 1920-х годах ученый побывал в экспедициях на Камчатке, Командорских островах, прошел по маршруту Советская Гавань — Хабаровск. Арсеньева интересовали самые разные темы: от борьбы с браконьерством до организации переписи населения. Его книги «По Уссурийскому краю» и «Дерсу Узала» были очень популярными у читателей.

Все эти годы Арсеньева, подполковника царской армии, не отпускало из поля зрения ГПУ. В 1931 году во владивостокской газете «Красное знамя» вышла статья «В. К. Арсеньев как выразитель идей великодержавного шовинизма».

Личный архив Арсеньева остался у его дочери, Натальи Владимировны Арсеньевой, у которой он был приобретен Приморским филиалом Географического общества СССР. После Великой Отечественной войны пропала его незаконченная рукопись «Страна Удеге», которую он писал 27 лет и редактировать которую согласился профессор Лев Штернберг. Эта рукопись не обнаружена до сих пор.

Владимир Арсеньев умер 4 сентября 1930 года от пневмонии, которой он заболел в экспедиции.

Именем Арсеньева названы ледник на северном склоне Авачинской сопки, гора в Приморском крае, приморский город, теплоход, самолет, перевалы в горном массиве Хибины Мурманской области и другие географические объекты.

*Информацию собрала ответственный редактор*
*Екатерина Осянина*

# СОДЕРЖАНИЕ

## ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

# ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

## Архитектурная переносимость оптимизаций C-циклов

Бабенков Юрий Михайлович, студент;
Молчанов Илья Аркадьевич, студент;
Лемза Данила, студент
Национальный исследовательский университет «Московский институт электронной техники» (г. Зеленоград)

*В работе представлена экспериментальная оценка переносимости относительного эффекта локальных компиляторных оптимизаций — автовекторизации (-ftree-vectorize / -fvectorize), разворачивания циклов (-funroll-loops) и выноса инвариантных ветвлений (-funswitch-loops) — при переносе четырёх тестовых циклов на языке C с архитектуры x86–64 на AArch64. Дополнительно исследована роль флага -fassociative-math в разблокировке векторизации циклов с аккумулятором. Установлено, что знак оптимизационного эффекта сохраняется между архитектурами, однако его абсолютная величина детерминирована в основном шириной SIMD регистров (256 бит AVX2 против 128 бит NEON).*

***Ключевые слова:*** *автовекторизация, разворачивание циклов, переносимость оптимизаций, GCC, LLVM/Clang, x86–64, AArch64, AVX2, SIMD.*

С ростом доли процессоров на архитектуре AArch64 в серверном (AWS Graviton, Ampere Altra), пользовательском (Apple Silicon) и встраиваемом сегментах перед разработчиками вычислительного ПО на языке C встаёт задача кроссплатформенного портирования критичных по производительности участков кода.

Компиляторы GCC и Clang предоставляют не только агрегированные уровни оптимизации (-O1, -O2, -O3), но и отдельные флаги, управляющие конкретными проходами (passes) оптимизирующего конвейера. При этом состав активных проходов различается: GCC активирует автовекторизацию (-ftree-vectorize) только начиная с -O3, тогда как Clang включает соответствующий проход (-fvectorize) уже на уровне -O2.

Цель настоящей работы — экспериментально установить, сохраняет ли изолированный компиляторный флаг свой относительный эффект при переносе с x86–64 на AArch64.

Таблица 1. **Исследуемые флаги оптимизации и их механизм действия**

| Флаг | Механизм действия |
|---|---|
| -ftree-vectorize / -fvectorize | Замена скалярных операций на SIMD-инструкции (AVX2: 8×float, NEON: 4×float) |
| -funroll-loops | Дублирование тела цикла для снижения накладных расходов на управление (cmp/branch) |
| -funswitch-loops | Вынос loop-invariant условия за пределы цикла с клонированием тела |
| -fassociative-math | Разрешение перестановки FP-операций (нарушение IEEE 754) для разблокировки параллельной редукции |

Отобраны четыре тестовых цикла, каждый из которых изолирует конкретный паттерн зависимости данных (табл. 2). Все массивы ограничены объёмом 4096 элементов × 4 байта = 16 КБ, что гарантирует полное размещение в кэше L1d (32–48 КБ) и превращает замер в compute-bound задачу [7].

Таблица 2. **Тестовые циклы и целевая оптимизация**

| Цикл | Паттерн | Целевой флаг |
|---|---|---|
| SAXPY | y [i] = a·x [i] + y [i] | -ftree-vectorize |
| Reduce | acc += a [i] | -fassociative-math |
| Copy | dst [i] = src [i] | -funroll-loops, -ftree-vectorize |
| Unswitch | if(mode) b [i]=a [i]*2<br>else b [i]=a [i]+1 | -funswitch-loops |

Хронометраж осуществляется вызовом clock_gettime(CLOCK_MONOTONIC) с наносекундной точностью. Каждый исполняемый файл запускался 200 раз. Основной метрикой является медиана, для каждого замера вычисляется 95 % доверительный интервал [6]: $CI_{95} = 1.96 \times \sigma / \sqrt{n}$. На диаграммах доверительный интервал обозначен горизонтальными усами.

Также для снижения уровня шума была строго зафиксирована тактовая частота ЦП и остановлены фоновые процессы.

Таблица 3. **Параметры тестовых стендов**

| Параметр | Стенд x86–64 | Стенд AArch64 |
|---|---|---|
| Процессор | Intel Core i5–8265U | Apple M1 |
| Базовая частота | 1.6 ГГц (зафикс.) | 2.064 ГГц |
| ISA / SIMD | x86–64, AVX2 (256 бит) | AArch64, NEON (128 бит) |
| Кэш L1d | 32 КБ, 8-way | 128 КБ, 8-way |
| ОС | Debian 12, ядро 6.1 | Fedora 43, ядро 6.18 (Asahi Linux) |
| GCC | 14.2 | 15.1 |
| Clang | 19.1 (LLVM) | 19.1 (LLVM) |
| Базовые флаги | -O2 -march=native | -O2 -march=native |

Для каждого цикла собрано 8 профилей флагов в двух компиляторах на двух архитектурах. Два компилятора использованы не для их сравнения, а для проверки устойчивости наблюдаемых закономерностей: если эффект флага воспроизводится и в GCC, и в Clang, он с большей вероятностью обусловлен природой трансформации, а не деталями реализации конкретного компилятора.

Автовекторизация (-ftree-vectorize / -fvectorize). Код тестового цикла:

```
for (int i = 0; i < n; i++)
    y[i] += a * x[i];
```

Цикл SAXPY — идеальный паттерн для автовекторизации: отсутствие зависимостей между итерациями, линейный доступ к памяти, единственная FMA-операция в теле.

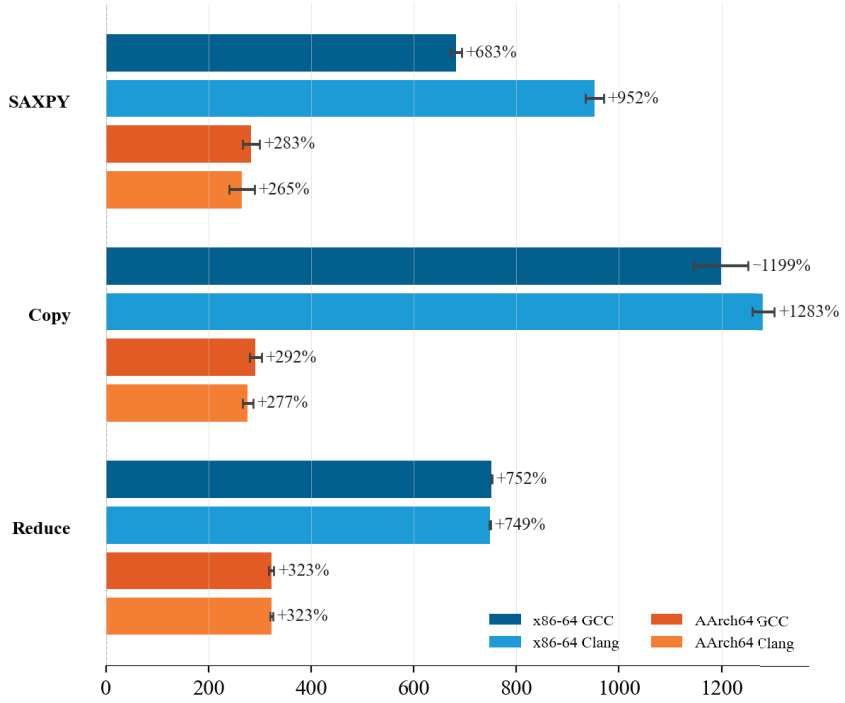


Рис. 1. **Ускорение от автовекторизации**

При -O2 GCC не выполняет автовекторизацию (+0 %). Явное включение -ftree-vectorize даёт +683 % (x86–64) и +283 % (AArch64). Clang при -O2 активирует -fvectorize автоматически, поэтому уже при базовых флагах демонстрирует +952 % (x86–64) и +265 % (AArch64).

Таблица 4. **Трансформация ассемблера SAXPY (GCC, x86–64)**

| Базовый уровень (скаляр) | vec_only (AVX2, 8×float) |
|---|---|
| vmovss xmm1,DWORD PTR [rdi+rax*1] | vmovups ymm1,YMMWORD PTR [rdi+rax*1] |
| vfmadd213ss xmm1,xmm0,DWORD PTR [rsi+rax*1] | vfmadd213ps ymm1,ymm2,YMMWORD PTR [rcx+rax*1] |
| vmovss DWORD PTR [rsi+rax*1],xmm1 | vmovups YMMWORD PTR [rcx+rax*1],ymm1 |
| add rax,0x4 | add rax,0x20 |
| cmp rdx,rax | cmp rax,rsi |
| jne | jne |
| vmovss xmm0,DWORD PTR [rsi] | mov eax,edx |

Таблица 5. **Трансформация ассемблера SAXPY (GCC, AArch64)**

| Базовый уровень (скаляр) | vec_only (NEON, 4×float) |
|---|---|
| ldr s31, [x0, x3] | ldr q4, [x0, x3] |
| ldr s30, [x1, x3] | ldr q3, [x1, x3] |
| fmadd s30, s31, s0, s30 | fmla v3.4s, v4.4s, v0.s[0] |
| str s30, [x1, x3] | str q3, [x1, x3] |
| add x3, x3, #0x4 | add x3, x3, #0x10 |
| cmp x2, x3 | cmp x3, x4 |
| b.ne | b.ne |

x86–64: скалярная vfmadd213ss (xmm, 1 float за такт) заменяется на vfmadd231ps (ymm, 8 float, шаг +0x20). AArch64: fmadd s30 заменяется на fmla v3.4s (Q-регистр, 4 float, шаг +0x10). Clang генерирует аналогичное преобразование на обеих платформах.

Ослабление ассоциативности (-fassociative-math). Код цикла Reduce:

```
float acc = 0.0f;
for (int i = 0; i < n; i++)
    acc += a[i];
```

Зависимость по аккумулятору блокирует векторизацию: стандарт IEEE 754 запрещает переупорядочивание FP-сложений [9]. Флаг -ftree-vectorize сам по себе не даёт эффекта (+0 %). Только -fassociative-math разрешает компилятору разбить аккумулятор на независимые частичные суммы, что разблокирует SIMD-параллелизм.

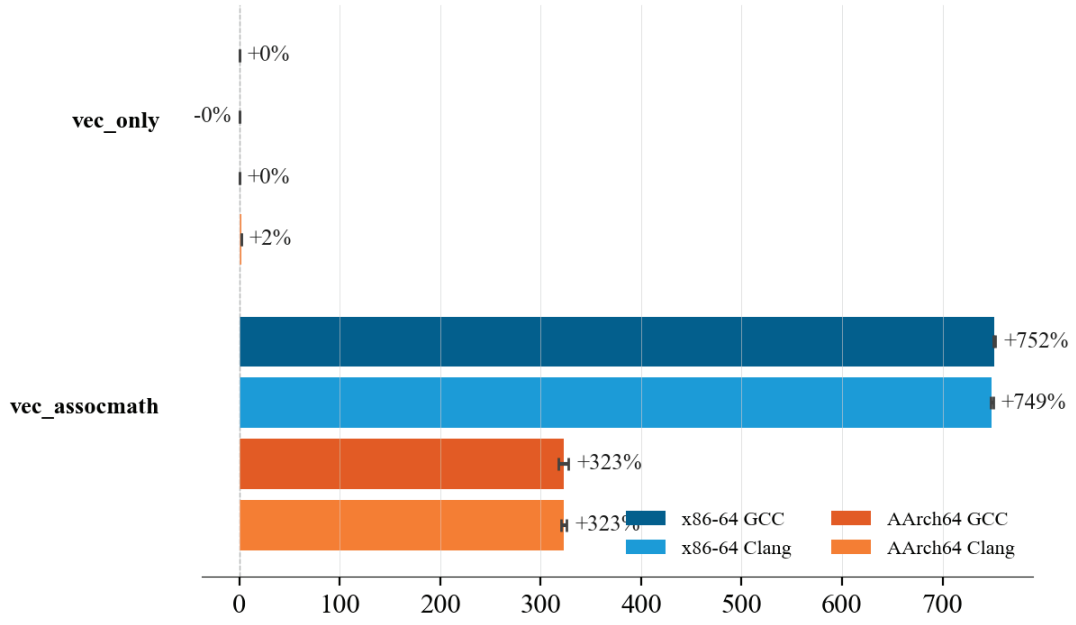


Рис. 2. **Влияние -fassociative-math на ускорение редукции**

Ускорение: +752 % (x86–64, GCC) и +323 % (AArch64, GCC) — близко к теоретическому пределу, определяемому шириной SIMD-регистра (8 и 4 float соответственно).

Таблица 6. **Трансформация ассемблера Reduce (GCC, x86–64)**

| Базовый уровень (скаляр) | vec_assocmath (AVX2) |
|---|---|
| vaddss xmm0,xmm0,DWORD PTR [rdi]<br>add rdi,0x4<br>cmp rdi,rax<br>jne<br>vmovss DWORD PTR [rsp-0x4],xmm0 | vaddps ymm0,ymm0,YMMWORD PTR [rax]<br>add rax,0x20<br>cmp rdx,rax<br>jne<br>vextractf128 xmm1,ymm0,0x1 |

Таблица 7. **Трансформация ассемблера Reduce (GCC, AArch64)**

| Базовый уровень (скаляр) | vec_assocmath (NEON) |
|---|---|
| ldr s30, [x0], #4<br>fadd s31, s31, s30<br>cmp x0, x1<br>b.ne | ldr q29, [x2], #16<br>fadd v0.4s, v0.4s, v29.4s<br>cmp x2, x3<br>b.ne |

Clang генерирует структурно идентичный цикл.

3.3. Разворачивание циклов (-funroll-loops). Исходный код цикла Copy:

```
for (int i = 0; i < n; i++)
    dst[i] = src[i];
```

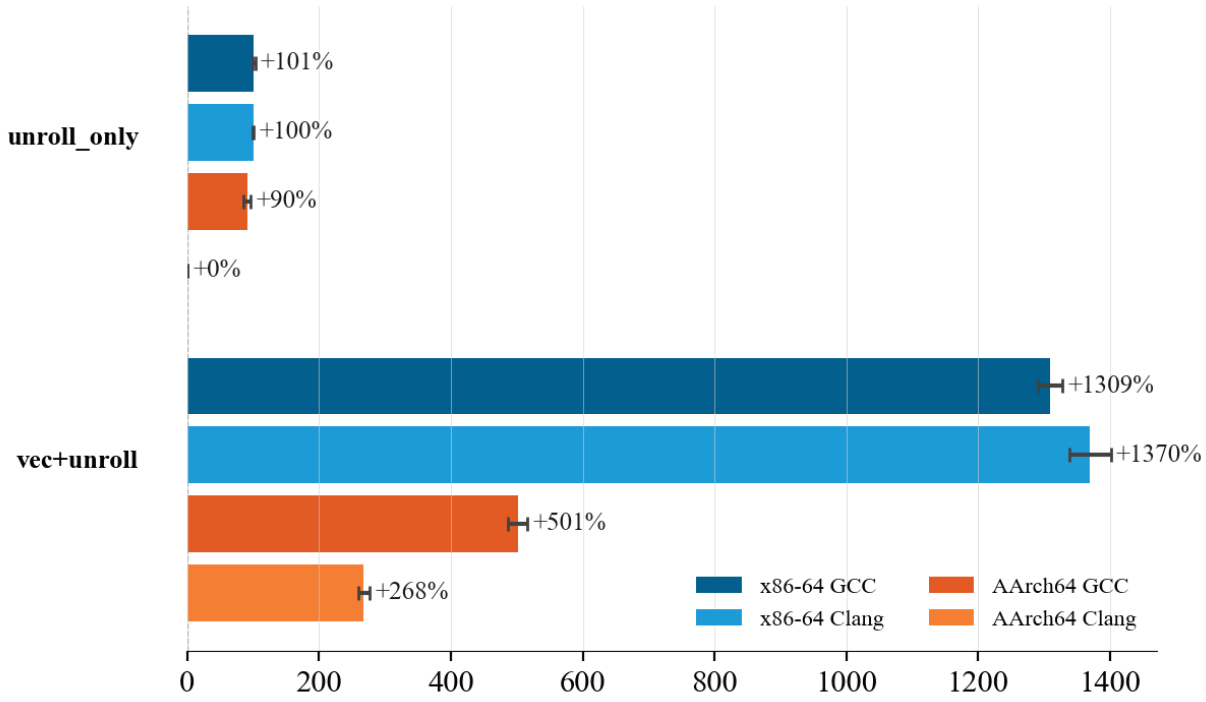


Рис. 3. **Ускорение Copy: разворачивание и комбинация vec+unroll**

Разворачивание без векторизации: GCC +101 % (x86–64), +90 % (AArch64). Комбинация vec+unroll: +1309 % (GCC x86–64), +501 % (GCC AArch64). Clang -funroll-loops на AArch64 не оказывает эффекта (+0 %), хотя на x86–64 ускорение составляет +100 %.

Таблица 8. **Трансформация ассемблера Copy (GCC, x86–64)**

| Базовый уровень | unroll_only | vec_only (AVX2) |
|---|---|---|
| vmovss xmm0,DWORD PTR [rdi+rax*1]<br>vmovss DWORD PTR [rsi+rax*1],xmm0<br>add rax,0x4<br>cmp rax,rdx<br>jne<br>vmovss xmm0,DWORD PTR [rsi] | je<br>vmovss xmm7,DWORD PTR [rdi+rax*1]<br>vmovss DWORD PTR [rsi+rax*1],xmm7<br>vmovss xmm8,DWORD PTR [rdi+rax*1+0x4]<br>vmovss DWORD PTR [rsi+rax*1+0x4],xmm8<br>vmovss xmm9,DWORD PTR [rdi+rax*1+0x8]<br>vmovss DWORD PTR [rsi+rax*1+0x8],xmm9<br>vmovss xmm10,DWORD PTR [rdi+rax*1+0xc]<br>vmovss DWORD PTR [rsi+rax*1+0xc],xmm10<br>vmovss xmm11,DWORD PTR [rdi+rax*1+0x10]<br>vmovss DWORD PTR [rsi+rax*1+0x10],xmm11<br>vmovss xmm12,DWORD PTR [rdi+rax*1+0x14]<br>vmovss DWORD PTR [rsi+rax*1+0x14],xmm12<br>vmovss xmm13,DWORD PTR [rdi+rax*1+0x18] | vmovups ymm0,YMMWORD PTR [rdi+rax*1]<br>vmovups YMMWORD PTR [rcx+rax*1],ymm0<br>add rax,0x20<br>cmp rax,rsi<br>jne<br>mov eax,edx |

Таблица 9. **Трансформация ассемблера Copy (GCC, AArch64)**

| Базовый уровень | unroll_only | vec_only (NEON) |
|---|---|---|
| ldr s31, [x0, x3]<br>str s31, [x1, x3]<br>add x3, x3, #0x4<br>cmp x3, x2<br>b.ne | ldr s6, [x0, x3]<br>str s6, [x1, x3]<br>ldr s7, [x0, x8]<br>str s7, [x1, x8]<br>ldr s16, [x0, x9]<br>str s16, [x1, x9]<br>ldr s17, [x0, x10]<br>str s17, [x1, x10]<br>...(8× ldr/str, шаг +0x20)<br>cmp x3, x2<br>b.ne | ldr q31, [x0, x3]<br>str q31, [x1, x3]<br>add x3, x3, #0x10<br>cmp x3, x4<br>b.ne |

Базовый уровень: скалярная загрузка vmovss / ldr s31 (4 байта). unroll_only: компилятор дублирует тело цикла, снижая накладные расходы на cmp/branch. vec_only: замена на vmovups ymm (32 байта, x86–64) / ldr q31 (16 байт, AArch64). Clang генерирует аналогичное преобразование.

Исходный код цикла Unswitch:

```
for (int i = 0; i < n; i++) {
    if (mode)
        b[i] = a[i] * 2.0f;
    else
        b[i] = a[i] + 1.0f;
}
```

Условие if(mode) инвариантно относительно итерационной переменной. Флаг -funswitch-loops клонирует цикл, проверяя условие однократно перед циклом, открывая каждую копию для автовекторизации.

GCC vec_only без unswitching: +0 % — ветвление внутри цикла блокирует векторизацию. Полная комбинация: +1224 % (x86–64), +390 % (AArch64). Clang выполняет unswitching автоматически при -O2, поэтому Clang vec_only: +1080 % (x86–64), +281 % (AArch64).

GCC unroll_only на x86–64: -38 % — разворачивание цикла с ветвлением без предварительного выноса увеличивает давление на I-cache и размножает точки предсказания [10].

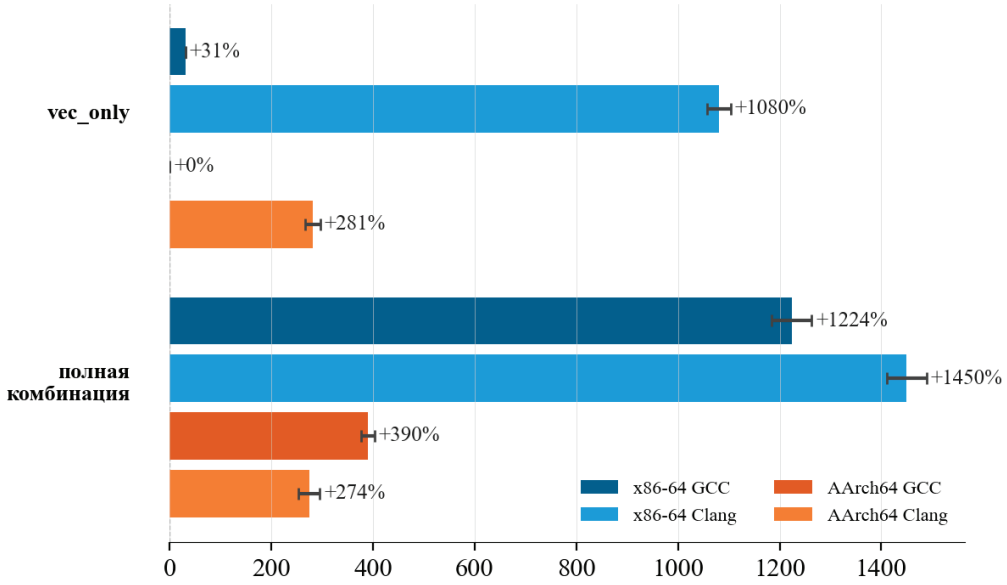


Рис. 4. **Ускорение Unswitch: vec_only vs полная комбинация**

Таблица 10. **Трансформация ассемблера Unswitch (GCC, x86–64)**

| Базовый уровень (ветвл.) | vec+unroll+unswitch (AVX2) |
|---|---|
| vmovss xmm0,DWORD PTR [rdi+rax*1] | je |
| vaddss xmm0,xmm0,xmm1 | vaddps ymm12,ymm4,YMMWORD PTR [rdi+rdx*1] |
| vmovss DWORD PTR [rsi+rax*1],xmm0 | vmovups YMMWORD PTR [rax+rdx*1],ymm12 |
| add rax,0x4 | vaddps ymm13,ymm4,YMMWORD PTR [rdi+rdx*1+0x20] |
| cmp rax,rdx | vmovups YMMWORD PTR [rax+rdx*1+0x20],ymm13 |
| jne | vaddps ymm14,ymm4,YMMWORD PTR [rdi+rdx*1+0x40] |
| vmovss xmm0,DWORD PTR [rsi] | vmovups YMMWORD PTR [rax+rdx*1+0x40],ymm14 |
| | vaddps ymm15,ymm4,YMMWORD PTR [rdi+rdx*1+0x60] |
| | vmovups YMMWORD PTR [rax+rdx*1+0x60],ymm15 |
| | vaddps ymm0,ymm4,YMMWORD PTR [rdi+rdx*1+0x80] |
| | vmovups YMMWORD PTR [rax+rdx*1+0x80],ymm0 |
| | vaddps ymm1,ymm4,YMMWORD PTR [rdi+rdx*1+0xa0] |
| | vmovups YMMWORD PTR [rax+rdx*1+0xa0],ymm1 |
| | vaddps ymm2,ymm4,YMMWORD PTR [rdi+rdx*1+0xc0] |

Таблица 11. **Трансформация ассемблера Unswitch (GCC, AArch64)**

| Базовый уровень (ветвл.) | vec+unroll+unswitch (NEON) |
|---|---|
| cbnz w3, .true_branch | ldr q21, [x0, x15] |
| fadd s0, s31, s30 | fadd v23.4s, v21.4s, v22.4s |
| add x3, x4, #0x4 | str q23, [x1, x15] |
| str s0, [x1, x4] | add x15, x15, #0x10 |
| cmp x3, x2 | ldr q24, [x0, x15] |
| b.eq .exit | fadd v31.4s, v24.4s, v22.4s |
| ldr s29, [x0, x3] | str q31, [x1, x15] |
| add x4, x4, #0x8 | add x15, x15, #0x10 |
| fadd s29, s29, s30 | ldr q0, [x0, x15] |
| str s29, [x1, x3] | fadd v30.4s, v0.4s, v22.4s |
| cmp x2, x4 | str q30, [x1, x15] |
| b.eq .exit | add x15, x15, #0x10 |
| ldr s31, [x0, x4] | |
| b .loop_top | |

x86–64: условный test/je внутри цикла устранён; вместо него — развёрнутый SIMD-цикл с vaddps ymm (8 float). AArch64: cbnz w3 вынесен перед циклом; NEON-цикл с fadd v23.4s (4 float). Clang генерирует аналогичный SIMD-код уже при vec_only.

**Заключение**

Проведённый эксперимент позволяет сформулировать следующие выводы:

1. Знак эффекта портируется. Если флаг ускоряет цикл на x86–64, он ускоряет его и на AArch64. Ни одного случая инверсии знака не обнаружено.
2. Масштаб эффекта не портируется. Ускорение от автовекторизации на x86–64 в 2–3 раза выше, чем на AArch64, что коррелирует с двукратной разницей в ширине SIMD-регистров (256 vs 128 бит).
3. -fassociative-math достаточен для разблокировки векторизации редукции; -ffast-math избыточен (побайтово идентичный ассемблер) [9].
4. Дополнительно проверен цикл с мультипликативной loop-carried зависимостью (acc = acc · 0.99 + a [i]): все исследованные флаги показали +0 % на обеих архитектурах, что подтверждает кроссплатформенную иммунность данного паттерна.
5. Вынос ветвлений — наименее переносимая по масштабу оптимизация.

**Ограничения и угрозы валидности**

Результаты получены на GCC 14.2/15.1 и Clang 19.1. Поведение проходов может измениться в будущих версиях. Четыре паттерна охватывают основные типы числовых циклов. Циклы с нерегулярным доступом к памяти (scatter/gather), зависимостями по индексу или полиморфными вызовами не рассматривались. Стенд x86–64 использует AVX2 (256 бит). На процессорах с AVX-512 абсолютные значения ускорений будут иными [4]. Аналогично для AArch64 с SVE/SVE2. Приводимые числовые значения характеризуют конкретные стенды. Направление эффектов обосновано структурой генерируемого ассемблера и воспроизведено на дополнительных процессорах.

Литература:


1. Callahan D., Dongarra J., Levine D. Vectorizing Compilers: A Test Suite and Results // Proc. ACM/IEEE Supercomputing. — 1988. — P. 98–105.
2. Maleki S., Gao Y., Garzarán M. J. et al. An Evaluation of Vectorizing Compilers // Proc. PACT. — IEEE, 2011. — P. 372–382.
3. Popov I., Chesnokov A. Comparison of Vectorization Capabilities of Different Compilers for X86 and ARM CPUs // arXiv:2502.11906. — 2025.
4. Шабанов Б. М., Телегин П. Н., Рыбаков А. А. Проблемы векторизации гнёзд циклов с использованием инструкций AVX-512 // Программная инженерия. — 2018. — Т. 9, № 5. — С. 203–213.
5. Fog A. Optimizing Software in C++. — Copenhagen, 2024. — URL: https://www.agner.org/optimize/optimizing_cpp.pdf.
6. Mytkowicz T. et al. Producing Wrong Data Without Doing Anything Obviously Wrong! // Proc. ASPLOS. — ACM, 2009. — P. 265–276.
7. Drepper U. What Every Programmer Should Know About Memory. — Red Hat, 2007. — 114 p.
8. Optimize Options // Документация GCC 14.2. — URL: https://gcc.gnu.org/onlinedocs/gcc/Optimize-Options.html.
9. Byrne S. Beware of fast-math. — 2021. — URL: https://simonbyrne.github.io/notes/fastmath/.
10. Branch predictor: How many "if"s are too many? // Cloudflare Blog. — 2025. — URL: https://blog.cloudflare.com/branch-predictor/.


## Автоматизация передачи данных из цифровых информационных моделей Revit в Renga

Бекишиева Мадина Абдулкеримовна, выпускник
Национальный исследовательский Московский государственный строительный университет

Информационное моделирование зданий (BIM) является ключевой технологией современного проектирования, обеспечивающей создание цифровых информационных моделей (ЦИМ) объектов капитального строительства. Одной из актуальных задач в области BIM является обеспечение эффективного взаимодействия между различными

программными платформами, в частности между Autodesk Revit и отечественной системой Renga [1].

Сейчас на российском рынке для проектных организаций возникает необходимость перехода на отечественное программное обеспечение, что связано с требованиями импортозамещения. При этом многие компании уже имеют накопленные проектные данные и библиотеки элементов в формате Revit. Проблема заключается в том, что традиционный обмен данными через формат IFC (Industry Foundation Classes) не всегда обеспечивает полную и корректную передачу параметрических свойств объектов [2].

Цель данной статьи — систематизировать существующие подходы к обмену данными между Revit и Renga, выделить их преимущества и ограничения. В статье рассматриваются формат IFC как универсальный стандарт обмена, нативные механизмы экспорта/импорта, а также решения, основанные на использовании открытых интерфейсов программирования API (Application Programming Interface) и промежуточных форматов данных.

Одним из основных способов обмена информационными моделями между разными BIM-программами является применение формата IFC, разработанного организацией buildingSMART. Этот формат представляет собой открытый стандарт описания строительных объектов и их характеристик, поддерживаемый большинством современных BIM-платформ [3].

Преимуществами IFC являются открытость формата, отсутствие привязки к конкретному вендору и возможность долгосрочного хранения данных. Однако при практическом использовании IFC для передачи моделей из Revit в Renga выявляются существенные ограничения, связанные с потерей части параметрической информации и неточным отображением сложных семейств и пользовательских элементов.

При экспорте моделей из Revit в IFC часть элементов передается в виде обобщенных тел без сохранения внутренних зависимостей и пользовательских параметров [4]. Часть объектов передается в виде непараметрических элементов с постоянными геометрическими параметрами [3]. При экспорте сложных семейств Revit в формат IFC теряется значительная часть параметрической информации, включая формулы зависимостей, пользовательские параметры и правила построения объектов.

В результате в целевой системе модель часто требует значительной ручной доработки. Кроме того, настройка карт соответствия классов IFC и категорий объектов в конкретной BIM-системе является трудоемкой задачей и зависит от версии программного обеспечения. Тем не менее, IFC остается важным инструментом для межплатформенного взаимодействия, особенно в случаях, когда требуется обмен данными между большим числом участников проекта и использованием различных программных продуктов.

Альтернативным подходом является использование API (Application Programming Interface) обеих систем для создания специализированного программного модуля передачи данных. Доступ к объектной модели через API позволяет программно считывать геометрию, параметры, атрибуты и взаимосвязи элементов, а затем воссоздавать их в целевой системе в соответствии с заданными правилами соответствия [1], [5]. Ключевым преимуществом такого подхода является возможность гибкой настройки процесса передачи данных под конкретные требования организации и типы проектов.

Также существуют решения, использующие промежуточные форматы данных (JSON, XML, собственные базы данных), где хранится структурированная информация об элементах модели. На этапе экспорта из Revit формируется такой файл, который затем анализируется модулем импорта в Renga. Между типами объектов двух систем создается библиотека соответствий, а для сложных элементов задаются специальные правила преобразования. Несмотря на большие первоначальные затраты на разработку, подобные решения позволяют существенно снизить трудоемкость миграции моделей и обеспечивают высокий уровень сохранности информации.

Перспективным направлением развития методов автоматизации передачи данных является интеграция BIM-платформ с системами управления жизненным циклом объектов и облачными хранилищами данных. В этом случае обмен моделями рассматривается не только как разовая миграция между программами, но и как элемент сквозного информационного пространства, включающего стадии проектирования, строительства и эксплуатации [3]. Для реализации таких сценариев требуется дальнейшее совершенствование форматов данных, развитие стандартов описания свойств объектов и унификация классификаторов.

Таким образом, анализ существующих подходов показывает, что ни один из методов обмена данными между Revit и Renga не является универсальным. Формат IFC обеспечивает наибольшую открытость и совместимость, но сопровождается потерей части параметрической информации. Нативные инструменты экспорта и импорта дают более точный результат, однако требуют строгой регламентации процесса моделирования. Решения, основанные на использовании API и промежуточных форматов, позволяют наилучшим образом учитывать особенности конкретной организации, но связаны с необходимостью разработки и сопровождения собственного программного обеспечения. Выявленные в ходе обзора особенности и ограничения методов обмена данных формируют теоретическую и методологическую основу для дальнейшей разработки собственной методики автоматизации передачи данных в рамках выпускной квалификационной работы.

Литература:

1. Сергеева Д. BIM-система проектирования Renga в тандеме с Revit // ИНФАРС. 2018. [Электронный ресурс] URL: https://infars.ru/blog/bim-sistema-renga-tandem-s-revit/ (дата обращения: 15.02.2026)

2. BIM-тандем Renga&Revit // Renga BIM. 2018. [Электронный ресурс] URL: https://rengabim.com/stati/bim-tandem-renga-revit/ (дата обращения: 16.02.2026)
3. Талапов В. В. Технология BIM: суть и особенности внедрения информационного моделирования зданий. М.: ДМК Пресс, 2019. 410 с.
4. Об интероперабельности информационных технологий в строительстве // Труды ИПУ РАН. 2018. [Электронный ресурс] URL: https://lab18.ipu.ru/projects/conf2018/3/26.pdf (дата обращения: 15.02.2026)
5. Нечаев А. А. Интероперабельность систем информационного моделирования зданий // 2023. [Электронный ресурс] URL: https://cyberleninka.ru/article/n/interoperabelnost-sistem-informatsionnogo-modelirovaniya-zdaniy (дата обращения: 16.02.2026)

# Синхронизация анимации 3D-аватара с потоковым синтезом речи на основе визем и эмоционального вектора VAD

Зубаиров Артур Маратович, студент
Российский государственный университет имени Косыгина А. Н. (Технологии. Дизайн. Искусство) (г. Москва)

*В статье рассматривается задача синхронизации движений губ и мимики 3D-аватара с речью, синтезируемой потоково по мере генерации ответа языковой моделью. Показано, что в конвейере с перекрытием генерации текста и синтеза речи аудио доставляется в браузер фрагментами, поэтому анимация не может ожидать полного аудиофайла ответа и должна собираться из отдельных фрагментов без слышимых или видимых разрывов на границах. Предложена архитектура, в которой временная шкала фонем извлекается из каждого синтезированного фрагмента независимо и передаётся на клиент вместе с аудио, а её интерполяция и смешение соседних визем выполняются на стороне рендеринга. Отдельно описан единый эмоциональный вектор, управляющий одновременно частотой моргания, скоростью анимации, наклоном головы и жёсткостью вторичной физики персонажа, что обеспечивает согласованность невербального поведения без отдельной настройки каждого канала анимации.*

***Ключевые слова:*** *синхронизация губ, визема, 3D-аватар, потоковый синтез речи, анимация в реальном времени, эмоциональная модель VAD.*

Визуальная синхронизация губ — один из факторов, определяющих, воспринимается ли 3D-персонаж как живой собеседник, а не как говорящая голова с произвольно двигающимся ртом [1]. В системах, где ответ языковой модели синтезируется в речь потоково (первое предложение озвучивается, пока генерируется следующее), аудио и данные для анимации поступают в браузер по частям — отдельными независимо синтезированными фрагментами по мере готовности. Например, если ответ состоит из трёх предложений, зритель услышит и увидит анимацию первого предложения ещё до того, как второе и третье вообще сгенерированы языковой моделью — к моменту готовности последнего фрагмента первый уже давно отыгран. Это меняет саму постановку задачи lip sync: вместо однократного сопоставления фонем с одним аудиофайлом требуется бесшовная анимация, собираемая из потока фрагментов, границы между которыми клиент не должен показывать зрителю.

Требуется способ формирования анимации рта 3D-аватара, удовлетворяющий трём условиям: (1) анимация конкретного фрагмента речи должна становиться доступной сразу по готовности этого фрагмента, без ожидания полного ответа; (2) переход между анимацией, порождённой соседними фрагментами, не должен создавать заметного скачка; (3) вычисление анимации на каждый кадр рендеринга должно быть дешёвым, поскольку выполняется на клиентском устройстве, а не на сервере. На рис. 1 показана общая схема конвейера: от генерации текста до анимации в браузере.

Существующие инструменты извлечения визем из аудио, включая Rhubarb Lip Sync [2], спроектированы для офлайн-обработки одного законченного файла целиком: они принимают на входе весь WAV-файл и возвращают полную временную шкалу визем за один вызов. Это хорошо работает для предзаписанной речи, но не рассчитано на ситуацию, когда в момент вызова доступен только очередной фрагмент ответа, а остальные ещё не существуют — инструмент как таковой не меняется, меняется то, как часто и над какими фрагментами он вызывается. В отличие от такого офлайн-подхода, предлагаемый метод рассматривает временную шкалу визем не как единый артефакт, готовый только по завершении всего ответа, а как поток независимых сегментов, каждый из которых обрабатывается и визуализируется сразу после синтеза соответствующего аудиофрагмента — это и устраняет необходимость ожидания полного завершения генерации ответа.

После синтеза аудио для очередного фрагмента текста WAV-файл передаётся тому же анализатору звуковой

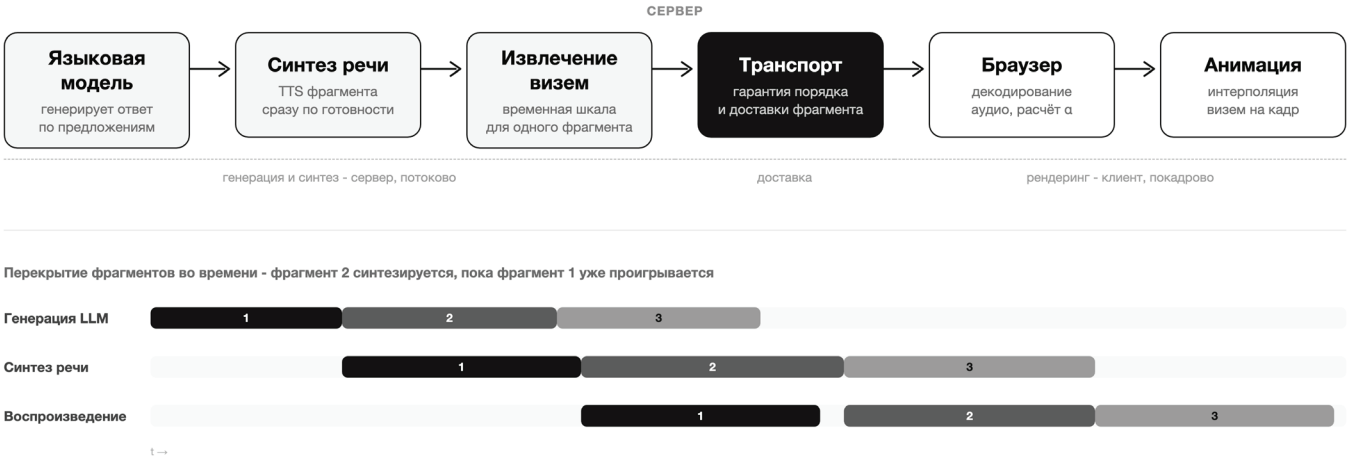


Рис. 1. **Потоковая сборка анимации: сервер обрабатывает фрагменты по мере готовности, транспорт гарантирует порядок и доставку, клиент рендерит покадрово; ниже показано перекрытие фрагментов 1–3 во времени**

волны, который возвращает временную шкалу визем (список пар «визема, время начала»), но уже только для этого фрагмента, а не для ответа целиком. Классификация ведётся по стандарту Престон Блэр [2], сокращающему многообразие фонем до девяти базовых визуальных форм рта: покой, смыкание губ, заднеязычные, шипящие, передние гласные, открытые гласные, лабиодентальные, зубные и альвеолярные согласные. Такое сокращение размерности принципиально для работы в реальном времени: девяти визуальных состояний достаточно для убедительной синхронизации, и они на порядок дешевле в вычислении и передаче, чем покадровая реконструкция формы рта.

Каждый фрагмент включает аудио в закодированном виде, массив визем с временными метками и идентификатор трека; всё это доставляется в браузер по мере готовности. Доставка идёт через постоянное соединение поверх той же инфраструктуры гарантированной доставки, что используется для остальных этапов конвейера: фрагменты не теряются и не приходят из очереди в перепутанном порядке. Поэтому на уровне анимации не нужно отдельно решать задачу упорядочивания или повторного запроса — она уже решена транспортным слоем, и анимация вправе полагаться на чистый, последовательный поток фрагментов. Клиент декодирует аудио через встроенный аудио-API и на каждом кадре рендеринга вычисляет смещение времени относительно начала воспроизведения текущего трека. Поскольку физиологически движение губ опережает звук, вычисленное смещение дополнительно корректируется на небольшую константу в сторону опережения. Так в анимацию закладывается известный эффект восприятия речи [1].

Прямое переключение между визуальными состояниями по границе временной метки визуально выглядит как дискретный скачок. Чтобы этого избежать, функция смешения на каждый момент времени t отвечает не одной готовой формой рта, а связкой из трёх чисел: какая визема сейчас, какая будет следующей, и насколько далеко мы продвинулись от одной к другой. Последнее — прогресс α — это просто доля пройденного пути между двумя соседними метками по времени: 0 сразу после смены визем, 1 прямо перед следующей. Формально: $\alpha = (t - t_0) / (t_1 - t_0)$, где $t_0$ и $t_1$ — метки времени текущей и следующей визем. Дальше 3D-рендерер сам плавно перетекает от одной формы рта к другой — технически это называется blend shape (веса, задающие, насколько сильно 3D-модель лица деформирована в сторону той или иной виземы), и рендерер смешивает веса двух соседних визем именно в пропорции α. Саму пару «текущая-следующая визема» ищет не полный перебор всех накопленных меток, а более быстрый бинарный поиск по отсортированному списку — грубо говоря, деление списка пополам на каждом шаге вместо проверки всех элементов подряд; на клиенте счёт визем может идти на сотни за диалог, и такой поиск не замедляется пропорционально их числу. Такая же логика смешения реализована дважды — на клиенте, работающем в браузере на TypeScript, и в переносимом виде на языке Rust для встраиваемых сценариев (десктопный клиент, плагин для стороннего программного обеспечения трансляции), где безопасность работы с памятью на уровне компиляции важна для стабильности процесса, работающего часами без перезапуска.

Помимо речи, персонаж должен независимо моргать, менять выражение лица и позу в такт эмоциональному тону реплики. Вместо настройки каждого канала анимации по отдельности все контроллеры читают единую структуру состояния. В её основе — трёхмерный эмоциональный вектор: валентность (насколько реплика позитивна или негативна по тону), возбуждение (насколько она энергична или, наоборот, спокойна) и доминирование (насколько уверенно или подчинённо звучит персонаж). Плюс вспомогательные параметры энергии, внимания и комфорта. Всё это формируется языковой моделью по ходу диалога. Наклон головы определяется осью доминирования; скорость анимации и физика вторичного движения — осью возбуждения; частота и характер моргания зависят от совокупного состояния. Такая архитектура устроена как реестр независимых контроллеров:

они читают общее состояние и исполняются поочерёдно на каждом кадре. Новое поведение добавляется реализацией одного интерфейса без изменения существующих контроллеров и без риска рассогласования между каналами, поскольку все они управляются одним и тем же источником эмоционального состояния.

Метод реализован для аватаров в формате VRM [3], рендеринг которых на клиенте выполняется средствами WebGL через библиотеку three-vrm [4]. Реализовано пять контроллеров анимации: моргание, взгляд, локомоция и эмоциональные жесты через конечный автомат, мимика, наклон головы и вторичная физика. Они работают независимо друг от друга поверх общей структуры эмоционального состояния и потока визем, поступающего вместе с аудио.

Корректность функции смешения (граница временной шкалы, середина интервала, момент после конца, отклонение некорректных данных при десериализации) проверена девятью модульными тестами, все проходят. Стоимость самого вычисления измерена микробенчмарком (criterion [5], 100 замеров на сценарий) на реалистичном фрагменте речи — таймлайне с визической плотностью около одной смены визем на 150 мс, что соответствует типичной выдаче анализатора на непрерывной речи. Результаты — в таблице 1.

Таблица 1. **Стоимость вычисления смешения визем (criterion, N = 100)**

| Сценарий | Время |
|---|---|
| Один вызов, фрагмент 3 с, 20 визем | 12,7 нс |
| Один вызов, фрагмент 8 с, 50 визем | 15,1 нс |
| Весь фрагмент 3 с при 60 кадр/с (180 вызовов) | 2,59 мкс |

Рост числа визем в 2,5 раза (с 20 до 50) увеличивает время одного вызова лишь на 19 % — это согласуется с логарифмической, а не линейной зависимостью от бинарного поиска. Обработка всех 180 кадров за проигрывание трёхсекундного фрагмента занимает 2,59 мкс совокупно — при бюджете в 16 700 мкс на один 60-кадровый цикл рендеринга — это менее 0,02 % бюджета, что подтверждает условие (3) числом, а не только словом «дёшево». Измерение охватывает исключительно саму функцию смешения в изоляции — оно не включает декодирование аудио, работу WebGL-рендерера или сетевую доставку фрагмента, и не является оценкой сквозной задержки всего диалогового конвейера.

Метод рассчитан на аватары с ограниченным, заранее известным набором визуальных выражений и не рассчитан на фотореалистичную реконструкцию формы рта по звуку — для такой задачи требуется принципиально иной подход на основе покадровой генерации изображения. Разделение ответственности между транспортным слоем и слоем анимации, описанное выше, работает только пока гарантии транспорта действительно соблюдаются: если развернуть тот же клиент поверх канала без гарантии доставки и порядка, придётся либо вернуть буферизацию и переупорядочивание на уровень анимации, либо мириться с редкими видимыми разрывами — метод не пытается быть устойчивым к обоим случаям одновременно, и это осознанный выбор границы, а не упущение.

Весовые коэффициенты сопоставления визем с фонемами подобраны по опубликованному стандарту [2] и не переоценивались экспериментально для конкретной реализации. Величина опережающей коррекции времени задана константой, а не выведена из измерений конкретной модели синтеза речи, — для голосов с иной артикуляционной динамикой она может потребовать пересчёта. Измерена стоимость только самого вычисления смешения визем в изоляции; сквозная задержка от готовности аудиофрагмента до отрисованного кадра на экране, включающая сеть, декодирование и рендеринг WebGL, отдельно не измерялась. Субъективная оценка убедительности синхронизации пользователями также не проводилась. Быстрое вычисление ещё не гарантирует, что зритель воспримет результат как достаточно плавный — это отдельный вопрос, и на него статья ответа не даёт.

Показано, что при потоковом синтезе речи задача синхронизации губ переходит от однократной обработки целого аудиофайла к бесшовной сборке анимации из независимо синтезированных фрагментов, а перенос ответственности за порядок и полноту потока на транспортный слой позволяет слою анимации оставаться простым. Предложенная архитектура реализована для 3D-аватаров и переносима между браузерным и встраиваемым окружением; корректность смешения визем подтверждена модульными тестами, а его вычислительная стоимость — микробенчмарком, показавшим менее 0,02 % бюджета одного кадра при 60 кадр/с.

Литература:

1. Miller M. R. Timing Matters: Effects of Response Delay on Perceived Naturalness in Robot Conversations: MSc project. — Oregon State University, 2025.
2. Wolf D. Rhubarb Lip Sync [Электронный ресурс]. — 2024. — Режим доступа: URL: https://github.com/DanielSWolf/rhubarb-lip-sync (23.07.2026)

3. VRM Consortium. VRM: A 3D Avatar File Format for VR Applications [Электронный ресурс]. — 2023. — Режим доступа: URL: https://vrm.dev/en/ (23.07.2026)
4. Pixiv Inc. three-vrm: VRM for Three.js [Электронный ресурс]. — 2024. — Режим доступа: URL: https://github.com/pixiv/three-vrm (23.07.2026)
5. Criterion.rs: Statistics-driven Microbenchmarking in Rust [Электронный ресурс]. — 2024. — Режим доступа: URL: https://github.com/bheisler/criterion.rs (23.07.2026)

# Подавление текстурных артефактов при сегментации венозного рисунка на RGB-изображениях с использованием морфологической фильтрации

Калюжнов Евгений Ярославович, студент;
Пономарёва Екатерина Александровна, студент
Нижегородский государственный технический университет имени Р. Е. Алексеева

*В работе рассматривается проблема подавления текстурных артефактов (волосяного покрова и татуировок) при сегментации вен на RGB-изображениях. Предложен комплексный подход на основе морфологических операций, включающий преобразование «чёрная шляпа», пороговую обработку и анализ в двух цветовых пространствах. Экспериментальная проверка на выборке из 1160 изображений подтвердила эффективность метода: точность сегментации в эталонной группе достигла 78,48 %. Выявлены ограничения метода и намечены пути их преодоления.*

***Ключевые слова:*** *компьютерное зрение, сегментация вен, морфологическая фильтрация, BlackHat преобразование, подавление артефактов, венепункция.*

## Введение

Венепункция — одна из самых массовых инвазивных процедур в здравоохранении (ежегодно более 450 млн манипуляций в РФ). Однако у 20–30 % пациентов вены плохо визуализируются из-за глубокого залегания, ожирения или особенностей кожи, что ведёт к многократным попыткам, осложнениям и увеличению времени приёма. Решением может стать система компьютерного зрения для автоматического поиска вен, но её точность сильно зависит от качества изображений. Ключевые текстурные артефакты — волосяной покров и татуировки — создают ложные структуры, снижающие эффективность алгоритмов. Задача данной работы — разработка и оценка методов морфологической фильтрации для подавления таких артефактов.

## Постановка проблемы

Анализ датасета из 1247 RGB-фотографий локтевой области выявил, что волосы образуют тонкие тёмные линии, по геометрии сходные с венами. Длинные контрастные волосы часто сохраняются после стандартной пороговой обработки, ошибочно классифицируясь как вены. Татуировки, особенно с тёмным пигментом, маскируют реальные сосуды или создают ложные контрастные структуры. Эксперименты показали, что наличие татуировок снижает точность определения точки пункции с 78,48 % (эталонная группа) до 65,75 %; сочетание татуировок с возрастными изменениями — до 34,68 %. Таким образом, требуется эффективный метод подавления этих артефактов на этапе предобработки.

## Методы подавления текстурных артефактов

В основе предлагаемого подхода лежит морфологическое преобразование «чёрная шляпа» (BlackHat):

**BlackHat(I) = Close(I) — I**,

где Close — морфологическое закрытие (дилатация + эрозия) с эллиптическим структурирующим элементом. Эта операция выделяет тёмные детали, размер которых меньше элемента, — именно вены. Для подавления волос применяется многоступенчатая фильтрация:

— открытие с ядром 3×3 удаляет точечный шум и короткие линии;

— закрытие с ядром 5×5 соединяет фрагменты вен и заполняет разрывы.

Для борьбы с татуировками используется комбинированный анализ в двух цветовых пространствах (HSV и YCrCb). Маска кожи строится как логическое пересечение пороговых масок в обоих пространствах, что снижает ложные срабаты-

вания на пигментированных участках. Параметры (размер ядра, порог бинаризации) подбираются автоматически методом Оцу, либо вручную через интерфейс.

**Результаты экспериментальной проверки**

Эксперимент проведён на выборке из 1160 изображений, разделённых на группы по наличию артефактов и возрасту. Результаты представлены в таблице.

Таблица 1. **Точность определения точки венепункции**

| Группа пациентов | Кол-во | Точность, % |
|---|---|---|
| Эталонная (до 70 лет, без артефактов) | ~700 | 78,48 |
| С татуировками/гематомами (до 70 лет) | 150 | 65,75 |
| Старше 70 лет (без артефактов) | 150 | 47,37 |
| Старше 70 лет + татуировки/гематомы | 100 | 34,68 |
| С ожирением (ИМТ > 30) | 100 | 42,38 |

Наилучшие результаты достигнуты в эталонной группе. Наличие татуировок и волос снижает точность, а их сочетание с возрастными изменениями — критически. Это подтверждает, что предложенные морфологические методы эффективны, но имеют ограничения.

**Обсуждение**

Совпадение динамических характеристик модели с расчётами подтверждает корректность настройки физики в Gazebo. Выявленное проскальзывание колёс при развороте (до 5 % ошибки одометрии) указывает на необходимость использования расширенного фильтра Калмана для комплексирования данных энкодеров, IMU и лидара в реальной системе. Отметим, что в большинстве аналогичных работ верификация ограничивается отдельными сценариями, тогда как наша модель проверена в комплексных условиях, приближенных к реальной эксплуатации.

Основные ограничения исследования: упрощение физических эффектов (деформация пола, работа редукторов, задержки связи). В перспективе планируется расширение симуляции на управление флотом роботов и интеграция с системой управления складом (WMS).

**Заключение**

В работе рассмотрена проблема подавления текстурных артефактов — волосяного покрова и татуировок — при сегментации венозного рисунка на RGB-изображениях. Предложен комплексный подход на основе морфологических операций, включающий преобразование «чёрная шляпа» (BlackHat), многоступенчатую фильтрацию и комбинированный анализ в двух цветовых пространствах.

Экспериментальная проверка на выборке из 1160 изображений подтвердила эффективность предложенного подхода: в эталонной группе пациентов точность сегментации достигла 78,48 %. Выявлены основные ограничения метода, связанные с длинными контрастными волосами и татуировками в области локтевого сгиба.

Дальнейшие исследования предполагают:

— разработку и обучение нейросетевого детектора артефактов для автоматического распознавания и удаления волос и татуировок;

— переход к мультимодальной архитектуре с использованием инфракрасной визуализации и ультразвукового сканирования для повышения точности на сложных группах пациентов;

— создание специализированного датасета изображений с различными типами текстурных артефактов для обучения моделей машинного обучения.

Литература:

1. Пономарёва Е. А. Разработка системы распознавания вен локтевой области для автоматизированного забора крови: ВКР. — НГТУ, 2026. — С. 16–17, 33–34, 54–59.
2. Гонсалес Р., Вудс Р. Цифровая обработка изображений. — 3-е изд. — М.: Техносфера, 2012. — 1104 с.
3. Samhitha N. et al. Advanced Image Segmentation and Machine Learning for Accurate Skin Disease Diagnosis // SSRN, 2025. DOI: 10.2139/ssrn.5229152.

4. Gopal M. S., Mallick S. P. Hair Removal Algorithm for Enhanced Vein Visualization in Near-Infrared Imaging // Springer, 2026. DOI: 10.1007/978–981–95–3492–0_4.
5. Ibsen M. et al. Face Beneath the Ink: Synthetic Data and Tattoo Removal with Application to Face Recognition // arXiv, 2023.

# Метод автоматической адаптации параметров морфологического анализа для выделения вен на RGB-изображениях с вариативным масштабом и разрешением

Калюжнов Евгений Ярославович, студент;
Пономарёва Екатерина Александровна, студент
Нижегородский государственный технический университет имени Р. Е. Алексеева

*В работе рассматривается проблема автоматической адаптации параметров морфологического анализа при выделении вен на RGB-изображениях с вариативным масштабом. Предложен метод автоматического подбора параметров на основе максимизации межклассовой дисперсии Оцу, адаптивной привязки размера ядра к характерному размеру изображения и итеративного перебора комбинаций. Экспериментальная проверка на 1160 изображениях подтвердила эффективность метода: точность сегментации в эталонной группе достигла 78,48 %.*

***Ключевые слова:*** *компьютерное зрение, сегментация вен, морфологический анализ, адаптация параметров, BlackHat, CLAHE, метод Оцу.*

## Введение

Системы компьютерного зрения для поиска вен на RGB-изображениях сталкиваются с проблемой вариативности условий съёмки: расстояние до руки, освещение, разрешение камеры. Толщина вены (3 мм в реальности) может занимать от 10 до 50 пикселей на снимке в зависимости от масштаба. Поскольку параметры морфологического анализа (размер ядра, порог) задаются в пикселях, фиксированные значения не могут быть оптимальны для всех случаев. Цель работы — разработка метода автоматической адаптации параметров для устойчивого выделения вен при вариативном масштабе и разрешении.

## Постановка проблемы

Анализ датасета из 1247 RGB-фотографий локтевой области показал, что масштаб съёмки варьируется от крупных планов (только ямка) до общих (вся рука). Также присутствует неравномерное освещение (блики, тени, перепады яркости). Глобальная эквализация гистограммы в таких условиях неэффективна — усиливает шум в тёмных областях и теряет детали в светлых. Необходим метод, автоматически подбирающий оптимальные параметры обработки для каждого изображения, компенсируя вариации масштаба и освещённости.

## Метод автоматической адаптации параметров

Предлагаемый метод основан на трёх ключевых механизмах адаптации: автоматическом подборе размера ядра морфологического оператора, адаптивной эквализации гистограммы с ограничением контраста (CLAHE) и автоматическом определении порога бинаризации методом Оцу.

## Адаптивный выбор размера ядра

Размер структурирующего элемента операции «чёрная шляпа» определяется на основе характерного размера изображения. Эмпирически установлено, что наилучшие результаты для выделения вен даёт эллиптический структурирующий элемент, размер которого составляет определённую долю от линейного размера изображения. Такой подход обеспечивает масштабную инвариантность: на изображениях, снятых крупным планом, ядро будет больше в абсолютных пикселях, чем на общих планах, но в обоих случаях оно будет соответствовать ожидаемой толщине вен. Для тонкой настройки размера ядра реализован итеративный перебор значений из диапазона от 9 до 61 пикселя.

## Адаптивная эквализация гистограммы (CLAHE)

Для компенсации неравномерного освещения применяется метод CLAHE (Contrast Limited Adaptive Histogram Equalization). В отличие от глобальной эквализации, CLAHE разбивает изображение на небольшие прямоугольные тайлы, в каждом из которых выполняется независимая эквализация гистограммы с искусственным ограничением максимального усиления контраста (clip limit). Коэффициент ограничения контраста автоматически подбирается из диапазона от 0.5 до 6.0 путём перебора комбинаций с различными размерами ядра.

**Автоматическое определение порога бинаризации**

Для преобразования полученной карты контрастности в бинарную маску вен применяется метод Оцу. Данный метод находит такое пороговое значение, которое максимизирует межклассовую дисперсию между двумя классами пикселей — фоном и венами. Для оценки качества каждой комбинации параметров используется вспомогательная функция, вычисляющая межклассовую дисперсию Оцу по гистограмме яркостей пикселей карты контрастности.

**Экспериментальная проверка**

Эксперимент проведён на 1160 изображениях, разделённых по группам. Результаты — в таблице 1.

Таблица 1. **Точность определения точки венепункции**

| Группа пациентов | Кол-во | Точность, % |
|---|---|---|
| Эталонная (до 70 лет, без артефактов) | ~700 | 78,48 |
| С татуировками/гематомами (до 70 лет) | 150 | 65,75 |
| Старше 70 лет (без артефактов) | 150 | 47,37 |
| Старше 70 лет + татуировки/гематомы | 100 | 34,68 |
| С ожирением (ИМТ > 30) | 100 | 42,38 |

Автоматический подбор параметров повысил устойчивость к вариациям масштаба и освещения по сравнению с фиксированными настройками. Основные ограничения — текстурные артефакты и низкий контраст в сложных группах.

**Обсуждение**

Ключевым преимуществом предложенного метода является его способность адаптироваться к широкому спектру условий съёмки без необходимости ручной настройки для каждого изображения. Автоматический подбор параметров на основе максимизации межклассовой дисперсии Оцу обеспечивает объективный критерий оптимальности, не зависящий от субъективной оценки оператора.

Однако метод имеет ограничения. При выраженных текстурных артефактах (волосяной покров, татуировки) межклассовая дисперсия может быть завышена за счёт ложных структур, что приводит к субоптимальному выбору параметров. Кроме того, для изображений с крайне низким контрастом (ожирение, глубокая пигментация) метод Оцу может не обеспечить надёжного разделения классов.

В перспективе планируется использование нейросетевых методов для предварительной классификации изображений по типу и автоматического выбора наиболее подходящего набора параметров, а также переход к мультимодальной архитектуре с использованием инфракрасной визуализации и ультразвукового сканирования.

**Заключение**

В работе предложен метод автоматической адаптации параметров морфологического анализа для выделения вен на RGB-изображениях с вариативным масштабом и разрешением. Метод включает адаптивный выбор размера ядра операции BlackHat на основе характерного размера изображения, автоматический подбор коэффициента ограничения контраста CLAHE и определение порога бинаризации методом Оцу с максимизацией межклассовой дисперсии.

Экспериментальная проверка на выборке из 1160 изображений подтвердила эффективность предложенного подхода: в эталонной группе пациентов точность сегментации достигла 78,48 %. Автоматическая адаптация параметров позволила существенно повысить устойчивость алгоритма к вариациям условий съёмки по сравнению с фиксированными настройками. Выявлены основные ограничения метода, связанные с текстурными артефактами и низким контрастом в сложных группах пациентов.

Дальнейшие исследования предполагают разработку нейросетевого классификатора для предварительной оценки типа изображения и выбора оптимального набора параметров, а также переход к мультимодальной архитектуре.

Литература:

1. Пономарёва Е. А. Разработка системы распознавания вен локтевой области: ВКР. — НГТУ, 2026. — С. 24–30, 33–37, 54–59.
2. Гонсалес Р., Вудс Р. Цифровая обработка изображений. — 3-е изд. — М.: Техносфера, 2012. — 1104 с.
3. CLAHE Documentation [Электронный ресурс] // Albumentations. — URL: https://albumentations.ai/docs/api_reference/augmentations/geometric/CLAHE (дата обращения: 19.07.2026).
4. Adaptive Histogram Equalization (CLAHE): Parameters and Tuning [Электронный ресурс] // Eureka Patsnap. — URL: https://eureka.patsnap.com/blog/adaptive-histogram-equalization-clahe (дата обращения: 19.07.2026).

5. Study on Automatic Segmentation Method of Retinal Blood Vessel Images // Journal of Computer Science, 2024. — URL: https://www.jsjkx.com (дата обращения: 19.07.2026).

# Использование контейнеризации для упрощения развертывания приложений

Козырев Петр Михайлович, студент
Московский государственный технологический университет «СТАНКИН»

*Контейнеризация снижает сложность развёртывания за счёт упаковки приложения, зависимостей и параметров запуска в переносимый образ. Цель статьи — рассмотреть, какие механизмы контейнеризации уменьшают различия между средами разработки, тестирования и эксплуатации. В работе анализируются контейнерные образы, Dockerfile, слои образа, реестр образов, Docker Compose, объект развёртывания Kubernetes, ограничения ресурсов и требования к безопасности контейнеров. Показано, что контейнеризация снижает число ручных операций при поставке приложения, но не отменяет управление конфигурацией, контроль версий образов, проверку уязвимостей и настройку оркестратора. Сделан вывод о том, что контейнеры полезны как стандарт упаковки и запуска, но сами по себе не обеспечивают надёжное развёртывание.*

***Ключевые слова:*** *контейнеризация, Docker, Dockerfile, OCI, Kubernetes, Docker Compose, контейнерный образ, реестр образов, CI/CD, развёртывание приложений.*

Развёртывание приложения часто осложняется различиями между локальной машиной разработчика, тестовым стендом и промышленной средой. На одном сервере может быть другая версия языка, иной набор системных библиотек, отсутствующая переменная окружения или зависимость, установленная вручную. В результате ошибка появляется не в логике приложения, а в окружении запуска. Контейнеризация решает эту проблему не за счёт изменения кода, а за счёт фиксации среды выполнения в виде образа, который можно собрать, сохранить в реестре образов и запустить на другой машине с совместимой средой выполнения контейнеров [1; 2].

Контейнер отличается от виртуальной машины тем, что обычно не содержит полноценной гостевой операционной системы. Контейнерный процесс использует ядро хостовой системы, а изоляция строится через механизмы Linux, включая пространства имён, cgroups (контрольные группы) и ограничения безопасности. Поэтому контейнер запускается быстрее и требует меньше ресурсов, чем виртуальная машина, но разделение между контейнером и хостом остаётся менее жёстким, чем при аппаратной виртуализации. Это различие важно для эксплуатации: контейнеры удобны для упаковки и масштабирования приложений, но требуют аккуратной настройки прав, сетей, томов и лимитов ресурсов [10; 12; 14].

Статья построена как обзор технических механизмов контейнерного развёртывания. В качестве основных источников выбраны официальные материалы Docker, Kubernetes и инициативы открытого стандарта контейнеров OCI, потому что они описывают формат образов, сборку, распространение, запуск контейнеров и управление рабочими нагрузками [1–13]. Академические публикации используются для сопоставления контейнеров с виртуальными машинами и для обоснования воспроизводимости вычислительных окружений [14; 15].

Основной артефакт контейнеризации — контейнерный образ. Образ содержит файловую систему приложения, команды запуска, метаданные и набор слоёв. Каждый слой появляется в результате инструкции сборки и может повторно использоваться при следующей сборке или загрузке [1]. Такой подход ускоряет поставку: если изменился только прикладной код, при повторной сборке и загрузке слои базового образа и зависимостей обычно переиспользуются. Для команды это означает, что развёртывание перестаёт быть набором ручных шагов на сервере и превращается в воспроизводимую сборку образа.

Dockerfile описывает процесс сборки образа в текстовом виде. В нём фиксируются базовый образ, установка зависимостей, копирование файлов, рабочая директория, переменные окружения и команда запуска [2]. Практическая ценность Dockerfile состоит в том, что он становится частью репозитория и проходит проверку вместе с кодом. Если команда меняет версию среды выполнения, системную библиотеку или способ запуска приложения, это изменение видно в истории версий. Контейнерный образ, собранный из такого файла, можно использовать в локальной разработке, автоматических тестах и промышленном развёртывании.

Многоэтапная сборка уменьшает размер образа для запуска. На первом этапе можно установить компилятор, зависимости разработки и собрать приложение, а на втором этапе перенести только результат сборки и минимальный набор файлов для запуска [3]. Это снижает объём передаваемых данных и уменьшает число инструментов внутри промышленного контейнера. Однако маленький образ не означает безопасный образ автоматически: базовый слой, системные библиотеки и зависимости приложения всё равно нужно обновлять и проверять сканерами уязвимостей [6].

Реестр образов выполняет роль хранилища и точки распространения образов. После сборки система непре-

рывной интеграции публикует образ с тегом или дайджестом содержимого, а серверы или кластеры загружают его при развёртывании [5; 13]. Здесь возникает важное эксплуатационное правило: тег вроде `latest` плохо подходит для повторяемой поставки, потому что со временем может указывать на другой образ. Kubernetes в своей документации рекомендует учитывать различие между тегами и дайджестами, так как дайджест однозначно связывает развёртывание с конкретным содержимым образа [7]. Для отката и расследования инцидентов такая точность важнее краткости тега.

Контейнеризация упрощает локальную разработку, когда приложение состоит из нескольких компонентов. Например, веб-сервису могут понадобиться база данных, брокер сообщений, кэш и отдельный рабочий процесс. Docker Compose позволяет описать эти сервисы, сети, переменные окружения и тома в одном файле [4]. Разработчик получает команду запуска всего окружения вместо набора инструкций по установке каждой зависимости. Такой подход особенно полезен для ввода нового участника в проект: он быстрее получает рабочую среду и меньше зависит от устных инструкций коллег.

В непрерывной интеграции и поставке контейнерный образ становится единым артефактом между стадиями конвейера поставки. Сначала система собирает образ, затем запускает тесты в окружении, близком к будущему запуску, затем публикует тот же образ в реестре и передаёт его в окружение эксплуатации. Это уменьшает риск ситуации, когда на тестовом стенде проверялся один набор зависимостей, а на сервер попал другой. Контейнеризация не устраняет все различия между средами: остаются конфигурация, секреты, сеть, файловые хранилища и права доступа. Но она фиксирует значительную часть среды выполнения и делает различия более явными.

Для промышленного развёртывания одного Docker или Compose обычно недостаточно, если приложение должно масштабироваться, обновляться без простоя и восстанавливаться после отказов. В этом случае используется оркестратор, например Kubernetes. Объект развёртывания описывает желаемое число экземпляров приложения, шаблон контейнерной группы, стратегию обновления и историю развёртываний [8]. При обновлении Kubernetes создаёт новые контейнерные группы с новым образом, постепенно заменяет старые и позволяет выполнить откат, если новая версия не проходит проверки или начинает ошибаться. Контейнер здесь выступает единицей поставки, а оркестратор управляет жизненным циклом этой единицы.

Упрощение развёртывания проявляется и в переносимости между инфраструктурами. Спецификация образа OCI задаёт формат образа, спецификация среды выполнения OCI — контракт запуска контейнерного процесса, а спецификация распространения OCI — протокол распространения образов через реестр [11–13]. Благодаря этим спецификациям один и тот же образ можно использовать в Docker и в других OCI-совместимых средах выполнения и платформах. На практике переносимость всё равно ограничивается архитектурой процессора, системными вызовами, драйверами, томами, сетевыми правилами и внешними сервисами, но стандартный формат образа уменьшает зависимость от одного инструмента.

Контейнеризация переносит часть сложности из ручной настройки сервера в описание образов и инфраструктуры. Если Dockerfile содержит устаревший базовый образ, запускает приложение от имени администратора или копирует секреты внутрь образа, контейнерная упаковка только закрепляет ошибку и распространяет её на все окружения. Поэтому промышленная практика должна включать минимальные базовые образы, многоэтапную сборку, запрет секретов в образах, подпись или проверку происхождения артефактов, регулярное обновление зависимостей и сканирование уязвимостей.

В Kubernetes дополнительно требуется управлять ресурсами. Для контейнерных групп и контейнеров задаются запрашиваемые ресурсы и предельные значения по центральному процессору и памяти [9]. Запрашиваемые ресурсы помогают планировщику выбрать узел, на котором хватит ресурсов, а предельные значения ограничивают потребление контейнера. Если эти параметры не заданы, один процесс может занять слишком много памяти или процессорного времени и повлиять на соседние рабочие нагрузки. Если они заданы без измерений, приложение может получить слишком мало ресурсов и начать падать при нормальной нагрузке. Поэтому контейнеризация упрощает упаковку, но не заменяет нагрузочное тестирование и наблюдаемость.

Безопасность контейнеров также зависит от настроек ядра и профилей изоляции. Kubernetes описывает применение профилей системных вызовов, AppArmor, SELinux, ограничений привилегий и запуска без прав администратора как часть защиты контейнерных рабочих нагрузок [10]. Эти механизмы ограничивают действия процесса даже в случае компрометации приложения. Однако они должны быть включены и согласованы с требованиями приложения. Контейнер, запущенный с широкими привилегиями, доступом к сокету Docker или прямым подключением каталога узла без ограничений, может создать риск для всего узла.

Контейнеризация упрощает развёртывание приложений, потому что превращает среду выполнения в версионируемый и распространяемый артефакт. Dockerfile фиксирует процесс сборки, слои образа ускоряют повторную поставку, реестр хранит версии артефактов, Compose описывает локальное многокомпонентное окружение, а Kubernetes управляет обновлениями и откатами в кластере. При этом контейнеры не решают сами по себе задачи безопасности, конфигурации, хранения данных и контроля ресурсов. Их следует рассматривать как основу для повторяемой поставки, которую нужно дополнять управлением секретами, проверкой образов, наблюдаемостью и дисциплиной версионирования.

Литература:

1. Docker Docs. Understanding the image layers. — 2026. — URL: https://docs.docker.com/get-started/docker-concepts/building-images/understanding-image-layers/ (дата обращения: 05.07.2026).
2. Docker Docs. Dockerfile overview. — 2026. — URL: https://docs.docker.com/build/concepts/dockerfile/ (дата обращения: 05.07.2026).
3. Docker Docs. Multi-stage builds. — 2026. — URL: https://docs.docker.com/get-started/docker-concepts/building-images/multi-stage-builds/ (дата обращения: 05.07.2026).
4. Docker Docs. Docker Compose. — 2026. — URL: https://docs.docker.com/compose/ (дата обращения: 05.07.2026).
5. Docker Docs. Image management. — 2026. — URL: https://docs.docker.com/docker-hub/repos/manage/hub-images/ (дата обращения: 05.07.2026).
6. Docker Docs. Hardened Docker Desktop: Minimal or distroless images. — 2026. — URL: https://docs.docker.com/dhi/core-concepts/distroless/ (дата обращения: 05.07.2026).
7. Kubernetes Documentation. Images. — 2026. — URL: https://kubernetes.io/docs/concepts/containers/images/ (дата обращения: 05.07.2026).
8. Kubernetes Documentation. Deployments. — 2026. — URL: https://kubernetes.io/docs/concepts/workloads/controllers/deployment/ (дата обращения: 05.07.2026).
9. Kubernetes Documentation. Resource Management for Pods and Containers. — 2026. — URL: https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ (дата обращения: 05.07.2026).
10. Kubernetes Documentation. Linux kernel security constraints for Pods and containers. — 2026. — URL: https://kubernetes.io/docs/concepts/security/linux-kernel-security-constraints/ (дата обращения: 05.07.2026).
11. Open Container Initiative. Image Specification. — 2026. — URL: https://github.com/opencontainers/image-spec (дата обращения: 05.07.2026).
12. Open Container Initiative. Runtime Specification. — 2026. — URL: https://github.com/opencontainers/runtime-spec (дата обращения: 05.07.2026).
13. Open Container Initiative. Distribution Specification. — 2026. — URL: https://github.com/opencontainers/distribution-spec (дата обращения: 05.07.2026).
14. Zhang Q., Liu L., Pu C., Dou Q., Wu L., Zhou W. A Comparative Study of Containers and Virtual Machines in Big Data Environment. — 2018. — URL: https://arxiv.org/abs/1807.01842.
15. Olaya P., Lofstead J., Taufer M. Building Containerized Environments for Reproducibility and Traceability of Scientific Workflows. — 2020. — URL: https://arxiv.org/abs/2009.08495.

## Методы обнаружения текста, сгенерированного искусственным интеллектом

Козырев Петр Михайлович, студент
Московский государственный технологический университет «СТАНКИН»

*Распространение больших языковых моделей сделало задачу обнаружения текста, сгенерированного искусственным интеллектом (ИИ), практической проблемой для образования, медиа, научных публикаций и корпоративного документооборота. Цель статьи — определить, какие группы методов применяются для выявления машинно-сгенерированного текста и почему результат такой проверки нельзя считать прямым доказательством нарушения правил использования ИИ или академической честности. Рассмотрены статистические признаки текста, стилометрия, классификаторы, методы на основе вероятностной кривизны, водяные знаки и устойчивость детекторов к перефразированию. Надёжность обнаружения зависит от языка, жанра, модели-генератора, способа редактирования текста и выбранного порога классификации. Поэтому детекторы следует использовать как инструмент оценки риска, а не как самостоятельное основание для обвинения автора.*

***Ключевые слова:*** *текст, сгенерированный ИИ, детекция текста, большие языковые модели, стилометрия, DetectGPT, водяные знаки, академическая честность, ложноположительные срабатывания.*

Задача обнаружения текста, сгенерированного ИИ, возникла как ответ на быстрое распространение систем, способных писать связные ответы, эссе, новости, отчёты и фрагменты научного текста. Для пользователя такой результат выглядит естественным, когда в нём выдержана тема, сохранён академический регистр, правильно расставлены термины и заметных грамматических ошибок нет. Поэтому простое экспертное чтение перестаёт быть

достаточным способом проверки. Возникает инженерная задача: найти признаки, по которым можно оценить вероятность машинного происхождения текста, не подменяя эту оценку окончательным суждением об авторстве.

Первый класс методов связан со статистическими характеристиками текста. GLTR как инструмент анализа рангов токенов показывает, что машинно-сгенерированные фрагменты часто используют слова, которые сама языковая модель считает наиболее вероятными в этом контексте. На этой идее строится визуальная и числовая оценка рангов токенов, энтропии и предсказуемости. Такой подход хорошо объясним: если текст слишком часто выбирает ожидаемые продолжения, он может отличаться от человеческого письма. Однако этот признак не универсален. Сильно отредактированный текст, специализированный жанр или формальная инструкция могут сделать человеческий текст таким же предсказуемым.

Второй подход опирается на стилометрию. В традиционной авторoведческой задаче анализируются частоты слов, длина предложений, пунктуация, синтаксические конструкции и устойчивые речевые привычки. Для ИИ-текста эти признаки тоже могут быть полезны, потому что модели часто сглаживают ритм, избегают резких тематических переходов и выбирают безопасные формулировки. Но исследование ограничений стилометрии для машинно-сгенерированных новостей показывает, что такой метод плохо работает как универсальный детектор. Он зависит от языка, жанра и того, какая именно модель использовалась для генерации.

Третий класс решений — обучаемые классификаторы. Они получают набор человеческих и машинных текстов, извлекают признаки или используют представления трансформерных моделей и затем выдают вероятность принадлежности фрагмента к одному из классов. Такой детектор можно адаптировать под конкретный жанр: академическое эссе, отзыв, новостной текст или ответ на экзамене. Цена этой адаптации — зависимость от обучающего набора. Если классификатор обучен на текстах одной модели, а проверяет тексты другой модели или тексты после ручной правки, качество обнаружения может заметно снизиться.

Методы семейства DetectGPT устроены иначе: они не требуют обучения отдельного классификатора на размеченном корпусе. DetectGPT использует идею вероятностной кривизны: текст, созданный языковой моделью, должен находиться в области, где небольшие перефразировки обычно уменьшают его вероятность. Fast-DetectGPT развивает эту идею и снижает вычислительные затраты. Этот подход позволяет проверять текст без подготовки большого набора примеров для каждого нового генератора. Но он требует доступа к языковой модели или близкому вероятностному оценщику, а качество зависит от того, насколько проверяемый текст похож на распределение, с которым работает этот оценщик.

Водяные знаки предлагают другой механизм. Во время генерации модель может немного смещать выбор токенов так, чтобы в тексте возникал статистически проверяемый след. Проверяющая сторона затем анализирует распределение токенов и оценивает наличие такого следа. В отличие от классификатора, водяной знак создаёт признак на этапе генерации, а не пытается восстановить авторство только по готовому тексту. Такой механизм работает только при двух условиях: генератор действительно встраивает знак, а проверяющий знает правило обнаружения. Кроме того, перефразирование, перевод или глубокая редактура могут ослабить след.

Перефразирование остаётся одним из главных способов обхода детекторов. Исследования показывают, что автоматическое или ручное переписывание может снижать качество классификаторов, методов семейства DetectGPT и водяных знаков. Это важно для практики: проверяемый текст редко бывает «чистым» результатом одного запроса к модели. Автор может попросить ИИ написать черновик, затем сократить его, добавить собственные фрагменты, заменить термины или прогнать через другую систему перефразирования. В таком случае задача обнаружения становится не бинарной, а смешанной: нужно оценить степень участия ИИ и характер последующей человеческой правки.

Для академической среды особенно опасны ложноположительные срабатывания. Работа о смещении GPT-детекторов против авторов, для которых английский не родной, показывает, что детекторы могут чаще ошибочно относить такие тексты к машинным. Механизм понятен: студент, пишущий на неродном языке, может использовать более простые синтаксические конструкции и предсказуемую лексику, что совпадает с признаками, на которые опирается часть детекторов. Поэтому результат автоматической проверки нельзя использовать изолированно. Он должен запускать дополнительное рассмотрение, а не заменять его.

Проблема осложняется языковой спецификой. Большая часть ранних работ и коммерческих инструментов ориентировалась на английский язык. Для русского языка нужны отдельные корпуса, потому что морфология, порядок слов, пунктуационные привычки и жанровые нормы отличаются. Задача RuATD показывает, что для русского языка требуется собственная оценка методов обнаружения искусственно сгенерированного текста, а не прямой перенос англоязычных результатов. Это особенно важно для образовательных организаций, где проверяются студенческие работы на русском языке и где ошибка классификации может повлиять на академическую репутацию.

Надёжная оценка детекторов требует специальных тестовых наборов. RAID предлагает рассматривать устойчивость к разным генераторам, видам атак, стратегиям выборки и жанрам. Такой подход ближе к реальной эксплуатации, чем тестирование на одном наборе «человек против ChatGPT». Детектор должен проверяться на текстах от моделей, не входивших в обучение, на перефразированных версиях, на смешанных текстах и на текстах разных авторских групп. Без такой проверки высокая точность на лабораторном наборе может создать ложное ощущение надёжности.

Практический процесс обнаружения ИИ-текста должен строиться как многоступенчатая проверка. На первом этапе автоматический детектор может выделить фрагменты с повышенным риском. На втором этапе эксперт оценивает жанр, стиль автора, историю правок, соответствие текста заданию и наличие источников.

Если риск остаётся высоким, проверка должна переходить к процедурным действиям: запросу пояснений автора, анализу черновиков, сравнению с промежуточными версиями или устной защите результата. Такой процесс медленнее, чем одноразовая проверка в автоматическом инструменте, но он снижает вероятность несправедливого решения.

Следовательно, методы обнаружения текста, сгенерированного ИИ, полезны только при правильном понимании их границ. Статистические признаки, стилометрия, классификаторы, DetectGPT-подходы и водяные знаки решают разные подзадачи и ломаются в разных условиях. Сильный детектор не доказывает нарушение, а даёт вероятностный сигнал, который нужно сопоставлять с контекстом создания текста. Поэтому в образовании, науке и документообороте такие методы должны использоваться вместе с конкретными правилами: раскрытием факта применения ИИ, хранением черновиков, допустимым уровнем машинной правки, процедурой апелляции и экспертной проверкой спорных случаев.

Литература:


1. Gehrmann S., Strobelt H., Rush A. M. GLTR: Statistical Detection and Visualization of Generated Text. — 2019. — URL: https://arxiv.org/abs/1906.04043.
2. Schuster T., Schuster R., Shah D. J., Barzilay R. The Limitations of Stylometry for Detecting Machine-Generated Fake News. — 2019. — URL: https://arxiv.org/abs/1908.09805.
3. Guo B., Zhang X., Wang Z., Jiang M., Nie J., Ding Y., Yue J., Wu Y. How Close is ChatGPT to Human Experts? Comparison Corpus, Evaluation, and Detection. — 2023. — URL: https://arxiv.org/abs/2301.07597.
4. Mitchell E., Lee Y., Khazatsky A., Manning C. D., Finn C. DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature. — 2023. — URL: https://arxiv.org/abs/2301.11305.
5. Bao G., Zhao Y., Teng Z., Yang L., Zhang Y. Fast-DetectGPT: Efficient Zero-Shot Detection of Machine-Generated Text via Conditional Probability Curvature. — 2023. — URL: https://arxiv.org/abs/2310.05130.
6. Kirchenbauer J., Geiping J., Wen Y., Katz J., Miers I., Goldstein T. A Watermark for Large Language Models. — 2023. — URL: https://arxiv.org/abs/2301.10226.
7. Krishna K., Song Y., Karpinska M., Wieting J., Iyyer M. Paraphrasing evades detectors of AI-generated text, but retrieval is an effective defense. — 2023. — URL: https://arxiv.org/abs/2303.13408.
8. Liang W., Yuksekgonul M., Mao Y., Wu E., Zou J. GPT detectors are biased against non-native English writers. — 2023. — URL: https://arxiv.org/abs/2304.02819.
9. Weber-Wulff D., Anohina-Naumeca A., Bjelobaba S., Foltýnek T., Guerrero-Dib J., Popoola O., Šigut P., Waddington L. Testing of Detection Tools for AI-Generated Text. — 2023. — URL: https://arxiv.org/abs/2306.15666.
10. Sadasivan V. S., Kumar A., Balasubramanian S., Wang W., Feizi S. Can AI-Generated Text be Reliably Detected? — 2023. — URL: https://arxiv.org/abs/2303.11156.
11. Liu Z., Yao Z., Li F., Luo B. On the Detectability of ChatGPT Content: Benchmarking, Methodology, and Evaluation through the Lens of Academic Writing. — 2023. — URL: https://arxiv.org/abs/2306.05524.
12. Dhaini M., Poelman W., Erdogan E. Detecting ChatGPT: A Survey of the State of Detecting ChatGPT-Generated Text. — 2023. — URL: https://arxiv.org/abs/2309.07689.
13. Yang L., Jiang F., Li H. Is ChatGPT Involved in Texts? Measure the Polish Ratio to Detect ChatGPT-Generated Text. — 2023. — URL: https://arxiv.org/abs/2307.11380.
14. Zaitsu W., Jin M. Distinguishing ChatGPT(-3.5, -4)-generated and human-written papers through Japanese stylometric analysis. — 2023. — URL: https://arxiv.org/abs/2304.05534.
15. Dugan L., Hwang A., Trhlik F., Ludan J. M., Zhu A., Xu H., Ippolito D., Callison-Burch C. RAID: A Shared Benchmark for Robust Evaluation of Machine-Generated Text Detectors. — 2024. — URL: https://arxiv.org/abs/2405.07940.
16. Shcherbakov A., Artemova E., Malykh V. RuATD 2022: Shared Task on Artificial Text Detection in Russian. — 2022. — URL: https://aclanthology.org/2022.bsnlp-1.26/.

# Методы снижения переобучения в глубоких нейронных сетях

Козырев Петр Михайлович, студент
Московский государственный технологический университет «СТАНКИН»

*Переобучение в глубоких нейронных сетях возникает, когда модель запоминает частные особенности обучающей выборки и хуже работает на новых данных. Цель статьи — рассмотреть методы, которые снижают этот риск при обучении глубоких нейронных сетей. Особое внимание уделяется штрафам за сложность модели, исключению нейронов, ранней остановке, расширению обучающей выборки, нормализации, сглаживанию меток и контролю качества на проверочной выборке. Показано, что отдельный приём редко решает проблему полностью: устойчивый результат дают согласованные настройки архитектуры, данных, функции потерь и процедуры проверки.*

***Ключевые слова:*** *глубокие нейронные сети, переобучение, регуляризация, исключение нейронов, расширение обучающей выборки, ранняя остановка, штраф за веса, проверочная выборка.*

Глубокая нейронная сеть способна находить сложные зависимости в данных, но та же способность создаёт риск переобучения. Если число параметров велико, а обучающая выборка ограничена или содержит шум, модель может подстроиться под случайные детали: редкие сочетания признаков, ошибки разметки, особенности измерительного оборудования или повторяющиеся шаблоны в наборе данных. На обучающих примерах такая модель показывает высокую точность, но при работе с новыми объектами ошибается чаще. Поэтому качество сети нужно оценивать не только по ошибке на обучении, но и по разрыву между обучающей и проверочной выборками.

Переобучение особенно заметно в задачах распознавания изображений, обработки естественного языка, медицинской диагностики и анализа временных рядов. В этих областях данные часто неоднородны: изображения снимаются разными устройствами, тексты принадлежат разным авторам, а измерения зависят от условий эксперимента. Если сеть обучалась на узком наборе примеров, она может принять несущественную деталь за устойчивый признак класса. Например, классификатор медицинских снимков может реагировать на отметку аппарата, а не на патологию, если такая отметка случайно совпадает с диагнозом в обучающей выборке. Такой результат опасен тем, что численно выглядит убедительно до проверки на независимых данных.

Первый способ снизить переобучение — ограничить сложность модели. Слишком глубокая или широкая сеть часто получает больше степеней свободы, чем нужно для конкретной задачи. Уменьшение числа слоёв, количества нейронов или размера векторных представлений снижает способность модели запоминать шум. Этот приём не требует сложной математики, но требует аккуратного подбора: чрезмерное упрощение приводит к недообучению, когда сеть не усваивает даже устойчивые закономерности. Практически это означает, что архитектуру следует выбирать через проверочную выборку, а не по максимальной точности на обучении.

Штраф за веса, часто называемый весовым затуханием, добавляет к функции потерь слагаемое, которое наказывает модель за слишком большие значения параметров. Идея проста: если сеть может решить задачу несколькими способами, предпочтение получает решение с меньшими весами и более гладкой зависимостью между входом и выходом. Такой подход снижает чувствительность к отдельным признакам и уменьшает вероятность запоминания случайных выбросов. В современных алгоритмах оптимизации штраф за веса часто отделяют от основной процедуры обновления параметров, потому что это даёт более предсказуемое поведение при адаптивных шагах обучения.

Исключение нейронов, или dropout, во время обучения случайно отключает часть нейронов и заставляет сеть не опираться на один узкий путь передачи сигнала. Каждый шаг обучения проходит как будто на немного изменённой сети, а при применении используется уже полная модель с пересчитанными весами. Такой механизм похож на усреднение множества частных моделей, но не требует хранить каждую из них отдельно. Польза метода особенно заметна в полносвязных слоях и больших сетях, где отдельные признаки могут начать чрезмерно влиять на ответ. Но долю отключаемых нейронов нельзя выбирать механически: слишком сильное исключение замедляет обучение и может ухудшить качество.

Ранняя остановка прекращает обучение до того, как сеть начнёт подстраиваться под шум. Для этого во время обучения выделяют проверочную выборку и после каждой эпохи сравнивают качество на обучающих и проверочных данных. Если ошибка на обучении продолжает снижаться, а ошибка на проверке перестаёт улучшаться или начинает расти, обучение останавливают и возвращают веса лучшей эпохи. Этот метод удобен тем, что не меняет архитектуру и почти не требует дополнительных вычислений. Его слабое место — зависимость от качества проверочной выборки: если она мала или плохо отражает будущие данные, остановка может быть преждевременной или запоздалой.

Расширение обучающей выборки снижает переобучение за счёт новых вариантов исходных примеров. В задачах обработки изображений применяют повороты, сдвиги, изменение яркости, масштабирование, кадрирование и добавление шума. В задачах анализа звука используют изменение темпа, высоты, громкости и фо-

новых помех. Такой подход учит модель игнорировать несущественные изменения и сохранять внимание на признаках, связанных с целевым классом. Но преобразования должны соответствовать предметной области: поворот цифры «6» на 180 градусов меняет её смысл, а небольшое изменение яркости дорожного знака обычно не меняет класс.

Смешивание обучающих примеров, известное как mixup, создаёт новый пример из двух исходных объектов и одновременно смешивает их метки. Вместо жёсткого требования «это ровно один класс» сеть получает более плавную задачу и учится вести себя устойчиво между наблюдаемыми точками данных. Такой метод снижает резкие границы решения и уменьшает уверенность модели в сомнительных областях пространства признаков. Он полезен, когда классы имеют плавные переходы или когда выборка ограничена. Но для задач, где смешение объектов лишено физического смысла, метод требует осторожной проверки.

Пакетная нормализация стабилизирует распределение промежуточных значений в слоях сети. Во время обучения она нормирует активации внутри мини-пакета и затем применяет обучаемое масштабирование и сдвиг. Первоначально метод предлагался как способ ускорить обучение глубоких сетей, но на практике он также может снижать переобучение за счёт дополнительного шума, возникающего при оценке статистик по мини-пакету. Нормализация не заменяет регуляризацию полностью, потому что её действие зависит от размера мини-пакета, архитектуры и режима применения модели. Тем не менее она часто улучшает устойчивость обучения и делает подбор скорости обучения менее хрупким.

Сглаживание меток уменьшает чрезмерную уверенность модели. Вместо целевой метки, где правильный класс получает вероятность 1, а остальные 0, модель получает немного распределённую цель: правильный класс остаётся главным, но другие классы получают малую долю вероятности. Это полезно, когда разметка может содержать ошибки или когда классы близки по смыслу. Сеть меньше стремится выдавать предельные вероятности и лучше калибрует свои ответы. Однако сглаживание меток может мешать задачам, где важна точная оценка различий между близкими классами, поэтому параметр сглаживания нужно выбирать по проверочным данным.

Снижение скорости обучения и корректное расписание её изменения тоже влияют на переобучение. Слишком большая скорость обучения мешает сети найти устойчивое решение, а слишком малая может привести к долгому запоминанию деталей обучающей выборки. На практике используют постепенное уменьшение скорости обучения, циклические расписания или остановку при отсутствии улучшения на проверке. Эти настройки не являются регуляризацией в узком смысле, но определяют траекторию оптимизации и качество найденного решения. Поэтому их рассматривают вместе с другими мерами защиты от переобучения.

Ансамбль моделей уменьшает зависимость результата от случайностей одного обучения. Несколько сетей обучаются с разной инициализацией, разными частями данных или разными архитектурами, а их ответы усредняются или объединяются голосованием. Ошибки отдельных моделей часто не совпадают полностью, поэтому общий ответ становится устойчивее. Недостаток ансамбля — рост вычислительных затрат при обучении и применении. В учебных и исследовательских задачах этот метод полезен как верхняя оценка достижимого качества, но в производственной системе его применяют только тогда, когда выигрыш оправдывает стоимость.

Контроль качества начинается с правильного разделения данных. Обучающая, проверочная и тестовая выборки должны быть независимыми по смыслу, а не только случайно разделёнными строками таблицы. Если несколько записей относятся к одному пользователю, пациенту, устройству или документу, их нельзя распределять между обучением и проверкой без учёта этой связи. Иначе модель фактически увидит часть будущих данных во время обучения, а оценка качества окажется завышенной. Для временных рядов особенно важно соблюдать порядок времени: будущие наблюдения не должны попадать в обучение при проверке прошлых решений.

Перекрёстная проверка помогает оценить устойчивость результата на малых выборках. Данные делят на несколько частей, модель несколько раз обучают на разных разбиениях и затем усредняют качество. Такой подход показывает, насколько вывод зависит от случайного выбора проверочной части. Для глубоких сетей полная перекрёстная проверка может быть дорогой, но её упрощённые варианты полезны при сравнении архитектур и настроек. Если разброс качества между разбиениями велик, это сигнал, что выборка мала, неоднородна или содержит нестабильные признаки.

Работа с шумной разметкой дополняет регуляризацию. Если часть меток ошибочна, сеть может выучить эти ошибки, особенно при долгом обучении и высокой мощности модели. Для снижения риска применяют очистку данных, повторную разметку спорных объектов, устойчивые функции потерь и проверку примеров с высокой ошибкой. В прикладных задачах это часто даёт больший эффект, чем усложнение архитектуры. Модель не может надёжно обобщать закономерности, если обучающие данные противоречат сами себе.

Выбор метода зависит от причины переобучения. Если данных мало, первым шагом обычно становится расширение выборки и строгая проверка разбиения. Если сеть чрезмерно сложна, помогают уменьшение архитектуры, штраф за веса и исключение нейронов. Если ошибка на проверке начинает расти после нескольких эпох, полезна ранняя остановка. Если модель слишком уверена в ответах, стоит проверить сглаживание меток и калибровку вероятностей. Такой причинный подход лучше, чем добавление всех методов сразу, потому что избыточная регуляризация может скрыть проблему данных или привести к недообучению.

Следовательно, снижение переобучения — не отдельная настройка, а система решений на всех этапах обучения. Необходимо контролировать состав выборок, мощность архитектуры, функцию потерь, процедуру оптимизации и способ оценки качества. Надёжная модель показывает не только высокую точность, но и устойчивое поведение на независимых данных, понятный разрыв между обучением и проверкой, приемлемую калибровку вероятностей и воспроизводимый результат при повторном обучении. В глубоких нейронных сетях именно сочетание регуляризации, аккуратной работы с данными и строгой проверки превращает высокую обучающую точность в практическую обобщающую способность.

Литература:

1. Zhang H., Cisse M., Dauphin Y. N., Lopez-Paz D. mixup: Beyond Empirical Risk Minimization. — 2017. — URL: https://arxiv.org/abs/1710.09412.
2. Shorten C., Khoshgoftaar T. M. A survey on Image Data Augmentation for Deep Learning. — Journal of Big Data. — 2019. — URL: https://journalofbigdata.springeropen.com/articles/10.1186/s40537–019–0197–0.
3. Ioffe S., Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. — 2015. — URL: https://arxiv.org/abs/1502.03167.
4. Muller R., Kornblith S., Hinton G. When Does Label Smoothing Help? — 2019. — URL: https://arxiv.org/abs/1906.02629.
5. Loshchilov I., Hutter F. Decoupled Weight Decay Regularization. — 2017. — URL: https://arxiv.org/abs/1711.05101.

## Технический долг в программных проектах: методы выявления и оценки

Козырев Петр Михайлович, студент
Московский государственный технологический университет «СТАНКИН»

*Технический долг в программном проекте возникает, когда команда принимает решение, ускоряющее текущую разработку, но увеличивающее стоимость будущих изменений. Цель статьи — рассмотреть методы выявления и оценки технического долга, применимые к коду, архитектуре, тестам и процессу сопровождения. В работе рассматриваются методы выявления и оценки технического долга: статический анализ кода, анализ признаков проблемного кода, метрики сопровождаемости, история изменений в системах контроля версий, данные трекеров задач, архитектурные зависимости и инструменты вроде SonarQube. Показано, что технический долг нельзя надёжно измерить одним числом: автоматические инструменты дают полезные сигналы, но требуют сопоставления с частотой изменений, критичностью модуля и стоимостью исправления. Сделан вывод о необходимости сочетать количественные метрики с инженерной экспертизой и приоритизацией работ по снижению долга.*

***Ключевые слова:*** *технический долг, качество программного обеспечения, статический анализ, признаки проблемного кода, SonarQube, SQALE, сопровождаемость, архитектурный долг, метрики кода.*

Технический долг описывает ситуацию, при которой быстрое проектное решение создаёт дополнительную стоимость будущих изменений. Эта метафора помогает различать осознанную отсрочку рефакторинга и накопление долга без явного решения. Иногда команда откладывает переработку кода, чтобы быстрее выпустить функцию и проверить гипотезу. Иногда долг появляется из-за отсутствия тестов, неясных границ модулей, копирования кода, временных обходов или устаревших зависимостей. В обоих случаях проблема проявляется позже, когда новое изменение требует больше времени, затрагивает больше файлов и чаще приводит к дефектам.

Выявление технического долга начинается с определения того, где именно он находится. Кодовый долг проявляется в сложных функциях, дублировании, длинных классах, нарушении инкапсуляции и неочевидных зависимостях. Тестовый долг связан с отсутствием автоматических проверок, хрупкими тестами и невозможностью безопасно менять поведение системы. Архитектурный долг возникает, когда компоненты зависят друг от друга так, что локальное изменение запускает цепочку правок. Процессный долг появляется в ручных релизах, незафиксированных договорённостях и отсутствии понятного владельца компонента.

Наиболее доступный способ первичного выявления долга — статический анализ кода. Такие инструменты не запускают программу, а проверяют исходный код по правилам: сложность методов, дублирование, нарушения стиля, потенциальные ошибки, небезопасные конструкции и проблемы сопровождаемости. SonarQube и похожие системы позволяют встроить эти проверки в конвейер поставки и получать повторяемый набор сигналов после каждого изменения. Практическая ценность статического анализа состоит в регулярности: он быстро

показывает новые проблемные места и не зависит от того, заметит ли их ревьюер.

При этом статический анализ не равен полному измерению технического долга. В работах Lenarduzzi et al. [5] показано, что для части правил SonarQube связь с дефектами и изменениями статистически значима, но эффект невелик. Другие исследования указывают, что разные инструменты обнаружения долга часто не совпадают в результатах [11; 12]. Это означает, что предупреждение анализатора следует рассматривать как индикатор, а не как доказательство высокой стоимости сопровождения. Если правило сработало в редко изменяемом вспомогательном модуле, риск может быть ниже, чем в коде, который команда правит каждую неделю.

Второй метод связан с признаками проблемного кода. К таким признакам относятся длинный метод, большой класс, цепочка условных операторов, временное поле, дублирование или чрезмерная связанность. Такой признак не доказывает дефект, но показывает место, где стоит проверить проектное решение. Для оценки технического долга эти признаки полезны тем, что переводят субъективное ощущение «код трудно менять» в более конкретные наблюдения. Однако решение о рефакторинге должно учитывать контекст: иногда длинная функция содержит линейный алгоритм и понятна, а короткая цепочка абстракций создаёт большую нагрузку на понимание.

Метрики кода дополняют признаки проблемного кода и статический анализ. Цикломатическая сложность показывает число независимых путей выполнения, размер модуля отражает объём кода, покрытие тестами показывает, какая часть строк, ветвей или функций исполняется тестами, а индекс сопровождаемости пытается объединить несколько признаков в одну оценку. Такие метрики удобны для отслеживания динамики: если сложность модуля растёт из релиза в релиз, а покрытие тестами падает, то команда получает ранний сигнал накопления долга. Но метрика сама по себе не объясняет причину. Она должна вести к анализу конкретного участка кода, а не заменять его.

История изменений в системе контроля версий помогает ранжировать долг по приоритету. Файл, который часто меняется, участвует в дефектах и одновременно имеет высокую сложность, создаёт больший риск, чем редко изменяемый устаревший модуль. Поэтому оценка технического долга должна учитывать частоту изменений, число авторов, плотность дефектов, связи между файлами и повторяемость правок. Такой подход особенно полезен для приоритизации: команда сначала снижает долг там, где он уже замедляет развитие продукта или повышает вероятность ошибок.

Системы управления задачами дают ещё один источник данных. Разработчики часто прямо фиксируют долг в комментариях, задачах и отложенных работах: «переписать после релиза», «убрать временное решение», «добавить тесты», «разделить модуль». Исследования явно признанного авторами технического долга показывают, что такие записи можно классифицировать и использовать как источник для планирования погашения. Преимущество этого метода в том, что он сохраняет причину отложенного решения. Недостаток состоит в неполноте: не каждый долг документируется, а часть комментариев устаревает.

Архитектурный долг требует отдельной оценки. Его трудно увидеть через одно правило статического анализа, потому что проблема находится не в строке кода, а в структуре зависимостей. Признаками архитектурного долга могут быть циклические зависимости, невозможность заменить компонент без изменения соседних модулей, общая база данных для разных подсистем, нарушение границ слоёв или постоянные изменения одного набора файлов при разных пользовательских задачах. Такой долг особенно опасен: он редко исправляется локальной правкой и часто требует серии согласованных изменений.

Оценка технического долга обычно описывается через основную сумму долга и проценты. Основная сумма — это стоимость исправления проблемного решения: рефакторинг, добавление тестов, разделение модуля, обновление зависимости или изменение архитектурного контракта. Проценты — это дополнительные затраты, которые команда несёт, пока долг не погашен: более долгие проверки кода, частые ошибки, сложная отладка, задержки релизов и осторожность при изменении кода. Для практического управления важны оба показателя. Если долг дёшев в исправлении, но дорог в сопровождении, его стоит погашать раньше, чем долг, который дорого исправить, но который редко влияет на работу.

Промышленные инструменты пытаются выражать долг в человеко-днях, рейтингах или индексах. Методология SQALE связывает найденные проблемы с оценкой времени на исправление и рассчитывает отношение долга к размеру проекта. Такой подход удобен для мониторинга, но опасен как единственный критерий. Если инструмент присваивает одинаковую стоимость двум нарушениям, он не знает, что одно находится в критичном модуле оплаты, а другое — в редко используемом отчёте. Поэтому числовая оценка должна дополняться бизнес-критичностью компонента и инженерной оценкой последствий.

Приоритизация работ по снижению долга должна учитывать количество найденных проблем, частоту изменений, критичность компонента и ожидаемую стоимость сопровождения. Практичный порядок может строиться так: сначала модули с частыми изменениями и высокой дефектностью, затем архитектурные зависимости, блокирующие развитие продукта, затем повторяющиеся признаки проблемного кода, которые мешают тестированию и ревью. Отдельно стоит выделять долг, связанный с безопасностью и устаревшими зависимостями, потому что его цена может проявиться не в скорости разработки, а в инциденте.

Выявление и оценка технического долга требуют сочетания нескольких источников данных. Статический анализ и метрики дают быстрый повторяемый сигнал. История изменений показывает, где долг уже влияет на развитие системы. Трекеры задач сохраняют причины отложенных решений. Архитектурный анализ выявляет зависимости, ко-

торые не видны в отдельных файлах. Наиболее надёжная оценка возникает тогда, когда эти данные сопоставляются с частотой изменений, критичностью модуля и стоимостью исправления. Поэтому технический долг следует рассматривать не как абстрактный показатель качества, а как управляемый риск будущих изменений.

Литература:


1. ГОСТ Р ИСО/МЭК 25010–2015. Системная и программная инженерия. Требования и оценка качества систем и программного обеспечения (SQuaRE). Модели качества систем и программных продуктов. — М.: Стандартинформ, 2015.
2. Fowler M. Technical Debt. — 2019. — URL: https://martinfowler.com/bliki/TechnicalDebt.html.
3. Fowler M. Code Smell. — 2006. — URL: https://martinfowler.com/bliki/CodeSmell.html.
4. Lacerda G., Petrillo F., Pimenta M., Guéhéneuc Y.-G. Code Smells and Refactoring: A Tertiary Systematic Review of Challenges and Observations. — 2020. — URL: https://arxiv.org/abs/2004.10777.
5. Lenarduzzi V., Saarimäki N., Taibi D. Some SonarQube Issues have a Significant but Small Effect on Faults and Changes. A large-scale empirical study. — 2019. — URL: https://arxiv.org/abs/1908.11590.
6. Lenarduzzi V., Saarimäki N., Taibi D. The Technical Debt Dataset. — 2019. — URL: https://arxiv.org/abs/1908.00827.
7. Lenarduzzi V., Besker T., Taibi D., Martini A., Arcelli Fontana F. Technical Debt Prioritization: State of the Art. A Systematic Literature Review. — 2019. — URL: https://arxiv.org/abs/1904.12538.
8. Molnar A.-J., Motogna S. Longitudinal Evaluation of Open-Source Software Maintainability. — 2020. — URL: https://arxiv.org/abs/2003.00447.
9. Molnar A.-J., Motogna S. Long-Term Evaluation of Technical Debt in Open-Source Software. — 2020. — URL: https://arxiv.org/abs/2007.13422.
10. Li Y., Soliman M., Avgeriou P. Identification and Remediation of Self-Admitted Technical Debt in Issue Trackers. — 2020. — URL: https://arxiv.org/abs/2007.01568.
11. Lefever J., Cai Y., Cervantes H., Kazman R., Fang H. On the Lack of Consensus Among Technical Debt Detection Tools. — 2021. — URL: https://arxiv.org/abs/2103.04506.
12. Pfeiffer R.-H., Lungu M. Technical Debt and Maintainability: How do tools measure it? — 2022. — URL: https://arxiv.org/abs/2202.13464.
13. Nayebi M., Cai Y., Kazman R., Ruhe G., Feng Q., Carlson C., Chew F. A Longitudinal Study of Identifying and Paying Down Architectural Debt. — 2018. — URL: https://arxiv.org/abs/1811.12904.
14. Robredo M., Saarimaki N., Taibi D., Penaloza R., Lenarduzzi V. Evaluating Time-Dependent Methods and Seasonal Effects in Code Technical Debt Prediction. — 2024. — URL: https://arxiv.org/abs/2408.08095.


## Методы репликации и шардинга в распределенных СУБД

Козырев Петр Михайлович, студент
Московский государственный технологический университет «СТАНКИН»

*Распределённые СУБД используют репликацию и шардинг для разных задач: репликация создаёт копии данных для отказоустойчивости и чтения, а шардинг делит данные между узлами для распределения объёма и нагрузки. Цель статьи — сравнить основные методы репликации и шардинга, применяемые в PostgreSQL, MySQL, MongoDB, Cassandra, ClickHouse, CockroachDB и YugabyteDB. В работе рассматриваются физическая и логическая репликация, схема с основным узлом и репликами, синхронный и асинхронный режимы, кворум, Raft, коэффициент репликации, выбор ключа шардинга, шардинг по диапазону и по хешу, перешардирование и проблема горячих шардов. Репликация и шардинг позволяют масштабировать систему только при правильном выборе ключей, правил кворума, стратегии отказа и модели согласованности. Сделан вывод о необходимости проектировать распределение данных исходя из профиля запросов, требований к задержке и допустимого риска устаревшего чтения.*

***Ключевые слова:*** *распределённые СУБД, репликация, шардинг, кворум, Raft, основной узел и реплики, MongoDB, Cassandra, ClickHouse, PostgreSQL, CockroachDB, YugabyteDB.*

Одна база данных на одном сервере ограничена ресурсами узла и становится точкой отказа. При росте нагрузки система должна отвечать на большее число запросов, хранить больше данных и продолжать работу при сбое машины или зоны доступности. Репликация и шардинг решают разные части этой задачи. Репликация со-

здаёт несколько копий одних и тех же данных, чтобы пережить отказ узла или перенести часть чтения на реплики. Шардинг распределяет разные части набора данных между узлами, чтобы один сервер не хранил и не обрабатывал весь объём.

Эти методы часто применяются вместе, но их нельзя смешивать. Репликация без шардинга повышает доступность, но не снимает ограничение на размер одной логической таблицы или пропускную способность записи на основном узле. Шардинг без репликации распределяет объём, но отказ одного шарда делает недоступной часть данных. Поэтому распределённая СУБД должна одновременно отвечать на два вопроса: сколько копий хранится у каждого фрагмента данных и по какому правилу данные попадают на конкретный шард.

Статья построена как обзор механизмов репликации и шардинга в современных СУБД. В корпус включены PostgreSQL и MySQL как распространённые реляционные СУБД со схемой с основным узлом и репликами, MongoDB и Cassandra как системы со встроенным распределением данных, ClickHouse как аналитическая СУБД с распределёнными таблицами, а также CockroachDB и YugabyteDB как распределённые SQL-системы, ориентированные на кворум — большинство реплик, достаточное для подтверждения операции [1–13]. Статья о Raft используется как базовый источник по консенсусу [14]. Сравнение проводится по четырём критериям: как выбирается узел записи, как подтверждается фиксация транзакции, как распределяются данные по шардам и что происходит при отказе или перераспределении нагрузки.

В схеме с основным узлом и репликами запись принимает один основной узел, а реплики получают изменения позже или в рамках протокола подтверждения. PostgreSQL поддерживает потоковую физическую репликацию на основе журнала предзаписи WAL: резервный сервер получает журнальные записи и воспроизводит их, а резервный сервер чтения может обслуживать запросы без изменения данных [1]. Такой подход подходит для резервирования и запросов только на чтение. Ограничение состоит в том, что запись всё равно проходит через основной узел, а при асинхронной передаче возможна потеря последних подтверждённых клиенту изменений при аварии основного узла.

Логическая репликация передаёт не физические страницы или журнал предзаписи как поток байтов, а изменения на уровне таблиц и строк. В PostgreSQL она строится через публикации и подписки: публикующий узел отдаёт изменения, а подписчик применяет их у себя [2]. Такой режим полезен для миграций, выборочной репликации таблиц, обмена между версиями и интеграционных сценариев. Ограничение такого режима — необходимость учитывать конфликты, ограничения схемы и поведение первичных ключей. Логическая репликация обычно хуже подходит как прозрачная замена физическому резервному серверу, но лучше подходит для выборочной передачи данных.

Синхронная и асинхронная репликация различаются моментом подтверждения записи. При асинхронной схеме основной узел может ответить клиенту до того, как реплика получила изменение. Это уменьшает задержку, но увеличивает окно потери данных при отказе. В синхронной репликации основной узел ждёт подтверждения от выбранных синхронных реплик; в полусинхронной схеме MySQL достаточно подтверждения о получении и записи событий в журнал передачи хотя бы одной репликой [3]. Это уменьшает риск потери данных, но добавляет сетевую задержку в путь записи. Поэтому выбор режима зависит от того, что важнее для конкретной системы: минимальная задержка записи или меньший риск отката последних операций.

В схемах с кворумом запись подтверждается после согласования с большинством реплик; в Raft большинство нужно для выбора лидера и фиксации записи в журнале. CockroachDB хранит диапазоны данных в нескольких репликах и использует Raft для согласования изменений [12; 14]. Такой подход позволяет автоматически переживать отказ меньшинства реплик, но требует связи между участниками кворума. Если сеть разделилась так, что ни одна часть не имеет большинства, запись должна остановиться, иначе система потеряет согласованность.

MongoDB использует набор реплик с основным узлом и вторичными узлами. Основной узел принимает запись, изменения попадают в журнал операций, а вторичные узлы применяют их асинхронно; при отказе основного узла участники набора реплик проводят выборы [5]. Это удобно для автоматического переключения и распределения чтения, но чтение с вторичного узла может вернуть устаревшие данные. Поэтому разработчик должен явно понимать предпочтение источника чтения, требование к подтверждению записи и риск задержки репликации.

Шардинг начинается с выбора ключа, по которому данные распределяются между узлами. Шардинг по диапазону делит данные по интервалам: например, заказы с идентификаторами от 1 до 1 000 000 попадают на один шард, следующие — на другой. Такой подход удобен для диапазонных запросов, но может создать горячий шард, если новые записи всегда попадают в конец диапазона. Шардинг по хешу применяет хеш-функцию к ключу и распределяет записи более равномерно. Он уменьшает риск перекоса записи, но усложняет диапазонные запросы, потому что соседние значения могут оказаться на разных узлах [6; 11].

MongoDB связывает шардинг с ключом шардинга. Если запрос содержит этот ключ или его префикс, система может направить операцию к нужному шарду. Если в условии нет ключа шардинга, запрос рассылается по нескольким шардам с последующим сбором результата [6]. Поэтому неправильный ключ шардинга не просто ухудшает баланс данных, а меняет стоимость запросов. Ключ должен учитывать кардинальность, распределение значений и типичные фильтры приложения.

ClickHouse использует движок распределённой таблицы для маршрутизации запросов к шарду и реплике внутри кластера [9]. В такой схеме локальные таблицы

хранят данные на узлах, а распределённая таблица выступает как логический вход для запроса. Ключ шардинга определяет, куда отправлять запись. При внутренней репликации запись отправляется в одну здоровую реплику каждого шарда; без неё данные отправляются во все реплики, а репликацию обеспечивает сама распределённая таблица. Для аналитических запросов это позволяет читать данные параллельно с нескольких шардов, но требует аккуратного выбора ключа, чтобы одна часть кластера не стала постоянным узким местом.

Cassandra проектирует распределение данных через ключ партиции и стратегию репликации на уровне пространства ключей [7]. Коэффициент репликации задаёт число копий, а стратегия сетевой топологии позволяет распределять реплики по дата-центрам. Такой подход подходит для высокой доступности и горизонтального роста, но требует проектировать таблицы исходя из запросов. Если ключ партиции выбран плохо, одна партиция может стать слишком большой или слишком горячей, а запросы без ключа партиции будут требовать дорогого обхода.

Со временем распределение данных меняется. Один шард может получить больше записей из-за популярного клиента, региона или временного диапазона. MongoDB использует фрагменты диапазона и балансировщик, а также поддерживает перешардирование коллекции на другой ключ шардинга [6]. YugabyteDB описывает таблетки как единицы шардинга и поддерживает шардинг по хешу и диапазону, автоматическое разделение таблеток и балансировку по кластеру [11]. Эти механизмы уменьшают ручную работу администратора, но не отменяют исходного выбора ключа: если поток вставок концентрируется вокруг одного значения ключа шардинга или одного диапазона, нужно перешардирование или смена ключа; один балансировщик этого не исправит.

Отказ узла в распределённой СУБД обрабатывается по-разному в зависимости от модели репликации. В системе с основным узлом и репликами нужно выбрать новый основной узел или вручную переключить приложение. В системе с кворумом запись продолжается, если доступно большинство реплик. В ClickHouse ReplicatedMergeTree репликация выполняется на уровне таблиц, а координатор, например ClickHouse Keeper, хранит метаданные реплик [8]. Важно, что отказоустойчивость зависит не от самого факта наличия копий, а от того, как система обнаруживает отказ, выбирает новый источник записи и предотвращает расхождение данных.

Сочетание репликации и шардинга даёт типовую схему: каждый шард хранит свою часть данных, а внутри шарда есть несколько реплик. Тогда потеря одной машины не уничтожает данные, а рост объёма можно обслуживать добавлением шарда. Ограничение такой схемы — сложность маршрутизации, миграции данных, межшардовых транзакций и согласованного чтения. Запрос, который раньше работал внутри одной таблицы, после шардинга может превратиться в распределённую операцию с несколькими узлами и разными задержками.

Главный вывод состоит в том, что репликация закрывает доступность и чтение, шардинг — объём данных и масштабирование записи, а горячая точка возникает на границе ключа и маршрутизации. Репликация создаёт копии данных, уменьшает риск потери при отказе и может переносить часть чтения на реплики, но добавляет задержку, выбор режима подтверждения и проблему устаревшего чтения. Шардинг распределяет объём данных и нагрузку между узлами, но требует правильного ключа шардинга, механизмов ребалансировки и контроля горячих шардов. Кворум и Raft помогают согласовать запись между репликами, но останавливают запись при потере большинства. Поэтому архитектура распределённой СУБД должна проектироваться от профиля запросов, требований к задержке, допустимой потери данных и правил восстановления после отказа.

Литература:


1. PostgreSQL Documentation. Log-Shipping Standby Servers. — 2026. — URL: https://www.postgresql.org/docs/current/warm-standby.html (дата обращения: 05.07.2026).
2. PostgreSQL Documentation. Logical Replication. — 2026. — URL: https://www.postgresql.org/docs/current/logical-replication.html (дата обращения: 05.07.2026).
3. MySQL 8.4 Reference Manual. Semisynchronous Replication. — 2026. — URL: https://dev.mysql.com/doc/refman/8.4/en/replication-semisync.html (дата обращения: 05.07.2026).
4. MySQL 8.0 Reference Manual. Group Replication. — 2026. — URL: https://dev.mysql.com/doc/refman/8.0/en/group-replication.html (дата обращения: 05.07.2026).
5. MongoDB Manual. Replication. — 2026. — URL: https://www.mongodb.com/docs/manual/replication/ (дата обращения: 05.07.2026).
6. MongoDB Manual. Sharding. — 2026. — URL: https://www.mongodb.com/docs/manual/sharding/ (дата обращения: 05.07.2026).
7. Apache Cassandra Documentation. Data Definition. — 2026. — URL: https://cassandra.apache.org/doc/latest/cassandra/cql/ddl.html (дата обращения: 05.07.2026).
8. ClickHouse Docs. Движки таблиц Replicated*. — 2026. — URL: https://clickhouse.com/docs/ru/engines/table-engines/mergetree-family/replication (дата обращения: 05.07.2026).

9. ClickHouse Docs. Движок Distributed. — 2026. — URL: https://clickhouse.com/docs/ru/engines/table-engines/special/distributed (дата обращения: 05.07.2026).
10. CockroachDB Docs. Architecture Overview. — 2026. — URL: https://www.cockroachlabs.com/docs/stable/architecture/overview.html (дата обращения: 05.07.2026).
11. YugabyteDB Docs. DocDB sharding. — 2026. — URL: https://docs.yugabyte.com/stable/architecture/docdb-sharding/ (дата обращения: 05.07.2026).
12. CockroachDB Docs. Replication Layer. — 2026. — URL: https://www.cockroachlabs.com/docs/stable/architecture/replication-layer (дата обращения: 05.07.2026).
13. YugabyteDB Docs. Raft consensus protocol. — 2026. — URL: https://docs.yugabyte.com/stable/architecture/docdb-replication/raft/ (дата обращения: 05.07.2026).
14. Ongaro D., Ousterhout J. In Search of an Understandable Consensus Algorithm. — 2014. — URL: https://raft.github.io/raft.pdf.

# Предиктивная аналитика технического состояния промышленного оборудования на основе сенсорных данных и методов глубокого обучения

Лихинин Александр Евгеньевич, студент магистратуры
Московский государственный технологический университет «СТАНКИН»

*Разработка метода предиктивной аналитики технического состояния промышленного оборудования, использующего сенсорные данные и глубокое обучение и позволяющего с высокой точностью прогнозировать остаточный ресурс и выявлять развивающиеся дефекты без ручного конструирования диагностических признаков.*

*Предложена гибридная нейросетевая архитектура, объединяющая одномерные свёрточные слои для извлечения локальных паттернов из многомерных временных рядов, двунаправленную долговременную краткосрочную память (BiLSTM) для моделирования долгосрочных зависимостей и механизм многоголового внимания для адаптивного взвешивания информативных участков сигнала. Входными данными служат нормализованные показания виброакустических, тепловых и режимных датчиков, сегментированные методом скользящего окна с оптимизированной шириной.*

*Валидация на открытом наборе данных C-MAPSS для турбореактивных двигателей продемонстрировала уменьшение среднеквадратичной ошибки прогноза остаточного ресурса на 13–18 % по сравнению с базовыми моделями LSTM и CNN, при этом F1-мера в задаче классификации дефектов подшипников на данных CWRU достигла 0,93. Модель сохраняла точность при переменных режимах нагружения.*

*Интеграция механизма внимания с глубокой рекуррентной обработкой «сырых» сигналов устраняет необходимость в эвристическом отборе признаков и повышает робастность прогностической системы к нестационарности эксплуатационных режимов. Предложенный подход может быть адаптирован к различным типам роторного оборудования.*

***Ключевые слова:*** *предиктивная аналитика, техническое состояние, остаточный ресурс, глубокое обучение, сенсорные данные, свёрточная нейронная сеть, долговременная краткосрочная память, механизм внимания.*

Промышленные предприятия эксплуатируют тысячи единиц мехатронного оборудования, оснащённого системами непрерывного мониторинга. Генерируемые потоки сенсорных измерений содержат информацию о постепенной деградации узлов, однако извлечение из них прогностических моделей сопряжено с трудностями, обусловленными многомерностью, нестационарностью и высоким уровнем шума [1, 2]. Традиционные подходы к оценке технического состояния опираются на эвристическое конструирование признаков (среднеквадратичные значения, пик-фактор, спектральные характеристики) и последующее применение методов машинного обучения, что ограничивает их обобщающую способность при смене режимов работы или типа оборудования.

Применение двунаправленных LSTM-сетей к задаче прогноза остаточного ресурса рассмотрено в [3]; авторы показали преимущество перед классическими рекуррентными моделями, однако не учли пространственную структуру многомерных сигналов. Свёрточные нейронные сети, адаптированные для анализа одномерных вибрационных рядов, описаны в [4] и демонстрируют высокую чувствительность к локальным аномалиям, но ограничены в захвате долговременных зависимостей. Гибридные CNN-LSTM архитектуры, частично устраняющие этот недостаток, предложены в [5], тем не менее их способность фокусироваться на наиболее информативных фазах деградации остаётся недостаточной. Внедрение механизмов внимания, первоначально разработанных для задач обра-

ботки естественного языка [6], в диагностические модели только начинает систематически изучаться.

Пусть имеется множество экземпляров оборудования, каждый из которых описывается последовательностью многомерных измерений, где число сенсорных каналов (вибрация, температура, давление, частота вращения и т. д.) обозначается как d, а длина жизненного цикла конкретного экземпляра до отказа — как T. Для каждого момента времени задана целевая функция остаточного ресурса, линейно убывающая от начального значения до нуля в момент отказа. В задаче классификации дефектов определён набор меток, соответствующих типу неисправности (например, норма, повреждение внутреннего кольца, наружного кольца, тел качения). Требуется построить модель, которая минимизирует среднеквадратичную ошибку прогноза остаточного ресурса и максимизирует F1-меру в задаче классификации, не прибегая к ручному отбору диагностических признаков.

Предлагаемая архитектура состоит из трёх последовательных блоков.

Первый блок — одномерные свёрточные слои с ядрами разного размера, применяемые параллельно к каждому из входных каналов. Операция свёртки выделяет локальные паттерны (всплески, тренды, переходные процессы), формируя карты признаков с сокращённой временной размерностью после объединения и числом фильтров, задающим глубину представления.

Второй блок — двунаправленная LSTM (BiLSTM), обрабатывающая последовательность карт признаков в прямом и обратном направлениях. Скрытое состояние для каждого временного шага агрегирует информацию о предшествующих и последующих состояниях, что принципиально важно для оценки степени износа, проявляющейся в долгосрочных трендах.

Третий блок — многоголовое внимание над выходами BiLSTM. Для каждого шага вычисляется контекстный вектор как взвешенная сумма скрытых состояний всех временных позиций; веса получаются посредством скалярного произведения запросов и ключей, масштабирования и нормализации функцией softmax.

Эксперименты выполнены на двух общедоступных наборах данных. Для прогнозирования остаточного ресурса использован набор C-MAPSS [7], содержащий данные о работе турбореактивных двигателей до отказа (поднаборы FD001–FD004, различающиеся числом режимов и типов неисправностей). Каждый экземпляр представлен 21 сенсорным каналом; длительности жизненных циклов варьируются от 128 до 362 отсчётов. Обучающая и тестовая выборки содержат по 100 экземпляров. Для классификации дефектов взяты сигналы виброускорения подшипников из репозитория CWRU [8] с четырьмя классами состояния: норма, повреждение внутреннего кольца, наружного кольца и тел качения. Данные разбиты в пропорции 70/15/15 для обучения, валидации и теста.

На поднаборе FD001, характеризующемся одним режимом работы, корень из среднеквадратичной ошибки прогноза остаточного ресурса для предложенной архитектуры CNN-BiLSTM-Attn составил 12,3. Для сравнения, базовая LSTM дала ошибку 15,1, одномерная CNN — 14,7, CNN-LSTM без внимания — 13,5, а модель Transformer — 13,0. Таким образом, разработанная модель сократила ошибку на 13–18 % относительно самых простых рекуррентных и свёрточных сетей. На более сложном поднаборе FD002 с шестью рабочими режимами абсолютные значения ошибок возросли, но относительное преимущество сохранилось: ошибка предложенной архитектуры составила 22,4 против 26,8 у Transformer, что свидетельствует о робастности к нестационарным условиям.

В задаче классификации дефектов подшипников модель достигла F1-меры 0,93, превзойдя CNN-LSTM (0,90) и чистый Transformer (0,91). Анализ матрицы весов внимания показал, что механизм внимания акцентируется на моментах входа тел качения в зону дефекта — наиболее информативных для диагностики участках сигнала.

Выполненное исследование подтверждает, что гибридная нейросетевая архитектура, сочетающая свёрточные, двунаправленные LSTM-слои и многоголовое внимание, обеспечивает более точный прогноз остаточного ресурса и классификацию дефектов по сравнению с типовыми глубокими моделями. Отказ от экспертного конструирования признаков упрощает перенос решения на оборудование различного типа. Дальнейшие работы целесообразно направить на включение в модель физических ограничений, интерпретацию весов внимания для формирования диагностических правил и адаптацию к потоковой обработке данных в реальном времени.

Литература:

1. Zhao R., Yan R., Chen Z., Mao K., Wang P., Gao R. X. Deep learning and its applications to machine health monitoring // Mechanical Systems and Signal Processing. — 2019. — Vol. 115. — P. 213–237.
2. Lei Y., Li N., Guo L., Li N., Yan T., Lin J. Machinery health prognostics: A systematic review from data acquisition to RUL prediction // Mechanical Systems and Signal Processing. — 2018. — Vol. 104. — P. 799–834.
3. Guo L., Li N., Jia F., Lei Y., Lin J. A recurrent neural network based health indicator for remaining useful life prediction of bearings // Neurocomputing. — 2017. — Vol. 240. — P. 98–109.
4. Ince T., Kiranyaz S., Eren L., Askar M., Gabbouj M. Real-time motor fault detection by 1-D convolutional neural networks // IEEE Transactions on Industrial Electronics. — 2016. — Vol. 63, № 11. — P. 7067–7075.
5. Chen Y., Peng G., Zhu Z., Li S. A novel deep learning method based on attention mechanism for bearing remaining useful life prediction // Applied Soft Computing. — 2020. — Vol. 86. — Art. 105919.

6. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser L., Polosukhin I. Attention is all you need // Advances in Neural Information Processing Systems. — 2017. — Vol. 30. — P. 5998–6008.
7. Saxena A., Goebel K. Turbofan Engine Degradation Simulation Data Set // NASA Ames Prognostics Data Repository. — 2008.
8. Case Western Reserve University Bearing Data Center. Seeded fault test data. — URL: https://engineering.case.edu/bearingdatacenter

# Сравнительный анализ форматов подготовки организаций к кризисам информационной безопасности

Мацкевич Владислав Викторович, студент
Московский государственный технологический университет «СТАНКИН»

*В статье рассматриваются основные форматы подготовки организаций к кризисам информационной безопасности: настольные учения, киберполигоны и компьютерные сценарные симуляции. Выполнено их качественное сравнение по критериям управленческой направленности, технического реализма, ресурсной ёмкости, повторяемости, измеримости результатов и гибкости сценариев. Показано, что каждый формат решает отдельный класс задач, а наибольший эффект достигается при последовательном сочетании организационных и технических упражнений.*

***Ключевые слова:*** *информационная безопасность, кризисное управление, настольные учения, киберполигон, сценарное моделирование, подготовка персонала, реагирование на инциденты.*

## 1. Введение

Управление киберинцидентом требует согласованных действий технических специалистов, руководителей, юристов, подразделений по связям с общественностью и владельцев бизнес-процессов. В отечественных нормативных и научных источниках реагирование рассматривается как последовательный процесс, включающий подготовку, обнаружение и регистрацию инцидента, локализацию, восстановление и анализ результатов [1–3]. Поэтому готовность организации определяется не только наличием средств защиты, но и способностью участников быстро распределять полномочия, обмениваться достоверной информацией и принимать решения при дефиците времени.

Для формирования такой готовности используются различные виды упражнений. Их прямое сопоставление затруднено, поскольку одни форматы ориентированы на коммуникацию и регламенты, а другие — на технические навыки. Цель статьи состоит в систематизации форматов подготовки и определении условий их рационального применения.

## 2. Основные форматы подготовки

Настольные учения представляют собой управляемое обсуждение кризисного сценария. Руководитель учения последовательно вводит новые обстоятельства, а участники определяют действия своих подразделений. В российской научной литературе такие упражнения рассматриваются как средство проверки регламентов, распределения ответственности, порядка эскалации и межведомственного взаимодействия [2–4]. Ограничением настольных учений является условность технической среды: заявленные действия не всегда подтверждаются практическим выполнением.

Киберполигон создаёт изолированную инфраструктуру, в которой воспроизводятся сети, сервисы, средства мониторинга и действия условного противника. Участники анализируют события, локализуют атаку и восстанавливают работоспособность систем. Российские исследования подчёркивают технический реализм и образовательную ценность киберполигонов, позволяющих безопасно исследовать атаки без воздействия на рабочую инфраструктуру [5]. Подготовка полигона требует оборудования, специалистов и значительного времени, а стратегические решения руководства могут оставаться вне технического упражнения.

Компьютерная сценарная симуляция занимает промежуточное положение. Она представляет кризис как последовательность событий и альтернативных решений, последствия которых рассчитываются или задаются сценарными правилами. Симуляция легко повторяется, позволяет фиксировать время и последовательность выбора, а также сравнивать результаты групп. При этом её реализм зависит от качества исходной модели и полноты учёта организационного контекста.

## 3. Критерии сравнительного анализа

Сравнение выполнено по шести критериям. Управленческая направленность характеризует пригодность формата для подготовки лиц, принимающих решения. Технический реализм отражает близость действий к рабочей инфраструктуре. Ресурсная ёмкость учитывает затраты на подготовку и проведение. Повторяемость показывает

возможность воспроизведения одинаковых условий. Измеримость связана с объективной фиксацией результатов, а гибкость сценария — со скоростью изменения вводных и состава участников.

Таблица 1. **Сравнение форматов подготовки**

| Критерий | Настольные учения | Киберполигон | Сценарная симуляция |
|---|---|---|---|
| Управленческая направленность | Высокая | Средняя | Высокая |
| Технический реализм | Низкий | Высокий | Средний |
| Ресурсная ёмкость | Низкая | Высокая | Средняя |
| Повторяемость | Средняя | Средняя | Высокая |
| Измеримость результатов | Средняя | Высокая | Высокая |
| Гибкость сценария | Высокая | Низкая | Высокая |

**4. Обсуждение результатов**

Результаты показывают, что универсального формата подготовки не существует. Настольные учения наиболее полезны для проверки полномочий, каналов коммуникации и решений, затрагивающих непрерывность бизнеса. Киберполигоны необходимы для подтверждения технической реализуемости планов и отработки взаимодействия специалистов. Сценарные симуляции удобны для регулярной подготовки, обучения большого числа участников и сопоставления альтернативных стратегий.

Рациональной является многоуровневая схема: настольное учение выявляет пробелы в регламентах, сценарная симуляция обеспечивает повторную отработку сложных развилок, а критически важные технические действия подтверждаются на киберполигоне. По итогам формируется отчёт с недостатками, ответственными лицами и сроками корректирующих мероприятий, что соответствует требованиям к анализу результатов реагирования [1, 2].

**5. Заключение**

Подготовка к кризисам информационной безопасности должна охватывать управленческие и технические компетенции. Настольные учения обеспечивают межфункциональную координацию, киберполигоны — практическую проверку действий, а компьютерные симуляции — повторяемость и измеримость обучения. Совместное применение форматов компенсирует ограничения каждого из них.

Литература:

1. ГОСТ Р 59712–2022. Защита информации. Управление компьютерными инцидентами. Руководство по реагированию на компьютерные инциденты. — М.: Российский институт стандартизации, 2022. — 20 с.
2. Петренко А. А., Петренко С. А. Киберучения: методические рекомендации ENISA // Вопросы кибербезопасности. — 2015. — № 3 (11). — С. 2–14.
3. Метельков А. Н. Киберучения: зарубежный опыт защиты критической инфраструктуры // Правовая информатика. — 2022. — № 1. — С. 51–60.
4. Козырь Н. С., Бэк В. С. Экосистема подготовки кадров в информационной безопасности: механизмы и перспективы // Правовая информатика. — 2025. — № 3. — С. 2–11.
5. Остапенко Г. А., Куликов С. С., Коноплин А. В., Остапенко А. А. Разработка архитектуры киберполигона для повышения качества учебного процесса // Информация и безопасность. — 2023. — Т. 26, № 1. — С. 101–108.

## Методы поддержки принятия управленческих решений при реагировании на инциденты информационной безопасности

Мацкевич Владислав Викторович, студент
Московский государственный технологический университет «СТАНКИН»

*В статье систематизированы методы поддержки управленческих решений при реагировании на инциденты информационной безопасности. Рассмотрены регламентные модели, многокритериальный анализ, вероятностные методы, оптимизационные и сценарно-экспертные подходы. Проведено их сравнение по интерпретируемости, требованиям к исходным данным, способности учитывать неопределённость, вычислительной сложности и пригодности для груп-*

*повой работы. Обоснована целесообразность комбинированного применения методов на различных этапах развития инцидента.*

***Ключевые слова:*** *реагирование на инциденты, поддержка принятия решений, многокритериальный анализ, экспертные оценки, вероятностная модель, оптимизация, информационный риск.*

## 1. Введение

Во время киберинцидента руководству приходится выбирать между альтернативами, каждая из которых связана с противоречивыми последствиями. Немедленное отключение системы может ограничить распространение атаки, но остановить критический бизнес-процесс; продолжение работы сохраняет доступность, однако увеличивает потенциальный ущерб. Решение принимается при неполной информации о масштабе компрометации, намерениях нарушителя и длительности восстановления.

ГОСТ Р 59712–2022 определяет основные стадии управления компьютерными инцидентами: обнаружение и регистрацию, реагирование и анализ результатов деятельности [1]. ГОСТ Р ИСО/МЭК 27005–2010 связывает выбор защитных мер с оценкой и обработкой риска информационной безопасности [2]. Однако нормативные документы задают общий процесс, а не универсальный алгоритм выбора между конфликтующими альтернативами. Цель статьи — сопоставить основные классы методов поддержки решений и определить границы их применимости.

## 2. Формализация задачи выбора

Пусть множество допустимых действий A включает изоляцию сегмента, блокировку учётных записей, переключение на резервную инфраструктуру, уведомление внешних сторон и другие меры. Каждое действие оценивается по набору критериев: ожидаемое снижение ущерба, время выполнения, стоимость, влияние на доступность, потребность в ресурсах, юридические последствия и обратимость. В простейшей многокритериальной модели интегральная оценка альтернативы определяется выражением $Q(a) = \Sigma\, w_i r_i(a)$, где $w_i$ — вес критерия, $r_i(a)$ — нормированная оценка действия. Такая модель делает логику выбора явной, но результат зависит от корректности весов и исходных оценок.

## 3. Основные методы поддержки решений

Регламентные методы основаны на заранее подготовленных планах реагирования, матрицах ответственности и сценарных инструкциях. Их преимущество — высокая скорость и понятность. Они эффективны для типовых событий, но плохо учитывают сочетание нескольких атак, неполноту данных и конфликт между безопасностью и непрерывностью бизнеса.

Многокритериальные методы сопоставляют действия по количественным и качественным показателям. В отечественной теории решений применяются взвешенная свёртка, парные сравнения, идеальная точка, последовательные уступки и множество Парето [3, 4]. Методы объяснимы, но чувствительны к субъективности оценок и выбору шкал.

Вероятностные методы связывают признаки атаки, состояния инфраструктуры и последствия ответных мер, уточняя риск по мере поступления данных. При оценке учитываются вероятность угрозы, уязвимость, ценность активов и ожидаемый ущерб [5]. Ограничения обусловлены дефицитом статистики и проверкой предположений.

Оптимизационные и сценарно-экспертные методы помогают распределять ограниченные ресурсы и анализировать последовательности событий. Сценарий учитывает качественные последствия, а оптимизационная модель формализует ограничения по времени, стоимости и ресурсам. Надёжность результата зависит от полноты модели и корректности допущений [3, 4].

## 4. Сравнительный анализ

Таблица 1. **Сравнение методов поддержки решений**

| Метод | Интерпретируемость | Учёт неопределённости | Требования к данным | Оперативность |
|---|---|---|---|---|
| Регламентный | Высокая | Низкий | Низкие | Высокая |
| Многокритериальный | Высокая | Средний | Средние | Средняя |
| Вероятностный | Средняя | Высокий | Высокие | Средняя |
| Оптимизационный / экспертный | Низкая–средняя | Средний | Высокие | Низкая–средняя |

**5. Комбинированное применение методов**

Выбор метода определяется стадией инцидента. В первые минуты применяются регламентные действия, после первичной оценки — многокритериальное сравнение вариантов. Вероятностные модели уточняют риск по мере поступления данных, а оптимизационные и сценарно-экспертные методы помогают распределять ресурсы. Результаты расчётов должны дополнять профессиональное суждение.

Практически значима двухконтурная схема: оперативный контур использует небольшой набор согласованных критериев, а аналитический — уточняет оценки и корректирует планы после инцидента. Каждая рекомендация должна сопровождаться объяснением критериев и качества исходных данных.

**6. Заключение**

Регламентные модели обеспечивают скорость, многокритериальный анализ — сопоставление целей, вероятностные методы — уточнение риска, а оптимизационные и сценарно-экспертные подходы — учёт ресурсов и вариантов развития событий. Наиболее устойчиво их комбинированное применение с учётом стадии инцидента и качества данных.

Литература:

1. ГОСТ Р 59712–2022. Защита информации. Управление компьютерными инцидентами. Руководство по реагированию на компьютерные инциденты. — М., 2022. — 20 с.
2. ГОСТ Р ИСО/МЭК 27005–2010. Информационная технология. Менеджмент риска информационной безопасности. — М.: Стандартинформ, 2011.
3. Ларичев О. И. Теория и методы принятия решений: учебник. — 2-е изд. — М.: Логос, 2002. — 390 с.
4. Орлов А. И. Теория принятия решений: учебник. — М.: Ай Пи Ар Медиа, 2022. — 826 с.
5. Милославская Н. Г., Толстой А. И. Управление рисками информационной безопасности: учебное пособие. — 3-е изд. — М.: Горячая линия — Телеком, 2023. — 224 с.

## Оценка зрелости процессов управления компьютерными инцидентами в организации

Мацкевич Владислав Викторович, студент
Московский государственный технологический университет «СТАНКИН»

*В статье рассматривается оценка зрелости процессов управления компьютерными инцидентами. Систематизированы организационные, процедурные, технические, кадровые и коммуникационные компоненты готовности. Предложена пятиуровневая шкала зрелости и набор показателей, основанных на документах, результатах учений и фактических сроках реагирования. Показано, что комплексная оценка позволяет выявлять критические разрывы и определять приоритеты совершенствования процесса.*

***Ключевые слова:*** *компьютерный инцидент, управление инцидентами, зрелость процесса, оценка готовности, показатели эффективности, киберустойчивость, информационная безопасность.*

**1. Введение**

Наличие технических средств защиты не гарантирует готовности организации к компьютерному инциденту. Даже при своевременном обнаружении атаки результат реагирования зависит от распределения полномочий, доступности специалистов, качества планов, устойчивости каналов связи и способности руководства принимать согласованные решения. Российская серия стандартов ГОСТ Р 59709–59712 рассматривает управление компьютерными инцидентами как непрерывную деятельность, включающую организацию процесса, обнаружение, регистрацию, реагирование и анализ результатов [1–4].

Формальное наличие документов не позволяет определить, способен ли установленный порядок работать в условиях реального кризиса. Для этого требуется оценивать зрелость процесса — степень его формализации, управляемости, измеримости и способности к совершенствованию. Цель статьи состоит в систематизации критериев такой оценки для внутреннего аудита и планирования развития.

**2. Компоненты готовности к реагированию**

Организационный компонент включает утверждённую политику управления инцидентами, распределение ролей,

полномочий и ответственности, порядок привлечения руководства и внешних организаций. Существенное значение имеет наличие постоянно актуализируемых контактных данных и назначенных заместителей для ключевых участников. ГОСТ Р 59711–2022 относит к организации деятельности формирование состава сил и средств, планирование, документирование и проведение тренировок [3].

Процедурный компонент характеризует полноту планов реагирования и наличие инструкций для типовых категорий инцидентов. Процедуры должны охватывать первичную оценку, классификацию, эскалацию, локализацию, сбор доказательств, восстановление и завершение работ. Их качество определяется не объёмом документа, а однозначностью действий, критериями перехода между стадиями и возможностью применения при неполной информации.

Технический компонент связан со способностью обнаруживать события, сохранять журналы, собирать сведения о затронутых активах, ограничивать распространение атаки и восстанавливать сервисы. Кадровый компонент отражает достаточность компетенций и доступность специалистов, а коммуникационный — готовность взаимодействовать с владельцами бизнес-процессов, юридической службой, контрагентами и государственными органами. Отдельным компонентом является организационное обучение: анализ завершённых инцидентов, проведение учений и контроль выполнения корректирующих мероприятий [4–6].

### 3. Уровни зрелости процесса

Для обобщённой оценки целесообразно использовать пятиуровневую шкалу от 0 до 4. Она позволяет перейти от бинарной проверки «есть или нет» к определению качества практического выполнения. Уровень устанавливается по документальным свидетельствам, результатам интервью, журналам событий, отчётам об учениях и фактическим данным о реагировании. Описание уровней приведено в таблице 1.

Таблица 1. **Уровни зрелости управления компьютерными инцидентами**

| Уровень | Характеристика | Основные признаки |
|---|---|---|
| 0. Отсутствующий | Процесс не определён | Действия зависят от отдельных сотрудников; роли, планы и показатели отсутствуют |
| 1. Реактивный | Выполняются разовые действия | Имеются отдельные инструкции, но порядок неполон и применяется непоследовательно |
| 2. Регламентированный | Процесс документирован | Назначены роли, утверждены планы, ведётся регистрация инцидентов и результатов |
| 3. Управляемый | Процесс измеряется и проверяется | Проводятся учения, контролируются сроки, анализируются отклонения и ресурсы |
| 4. Совершенствуемый | Процесс регулярно оптимизируется | Метрики используются для улучшений; уроки учитываются в планах, архитектуре и обучении |

Оценка проводится отдельно по каждому компоненту. Пусть $s_i$ принимает значение от 0 до 4, а $w_i$ отражает значимость компонента для конкретной организации. Тогда нормированный индекс зрелости может быть рассчитан как $M = (\Sigma w_i s_i)/(4\Sigma w_i)$. Значение M лежит в диапазоне от 0 до 1, однако итоговый показатель не должен скрывать критические провалы. Например, высокий общий результат не компенсирует отсутствие резервного канала связи или невозможность восстановить критическую систему. Поэтому вместе с индексом необходимо фиксировать минимальную оценку и перечень обязательных условий, невыполнение которых ограничивает общий уровень.

### 4. Показатели и порядок проведения оценки

Для снижения субъективности зрелость подтверждается количественными и качественными данными: временем регистрации, эскалации, локализации и восстановления; долей инцидентов, обработанных по утверждённому порядку; полнотой журналирования; результатами учений и выполнением корректирующих мероприятий. Оценка включает определение области проверки, сбор документов и фактических данных, выставление баллов по единой шкале и проверку результатов интервью либо разбором инцидента. Для каждого недостатка устанавливаются риск, ответственный и срок устранения. Самооценку целесообразно дополнять независимым аудитом, а показатели интерпретировать с учётом контекста [5, 6].

### 5. Заключение

Зрелость управления компьютерными инцидентами характеризует организационную, процедурную, техническую, кадровую и коммуникационную готовность. Пятиуровневая модель, измеримые показатели и анализ фактических действий позволяют выявлять критические разрывы и формировать последовательную программу развития. Регулярная оценка превращает реагирование из набора разовых действий в управляемый процесс.

Литература:

1. ГОСТ Р 59709–2022. Защита информации. Управление компьютерными инцидентами. Термины и определения. — М., 2022.
2. ГОСТ Р 59710–2022. Защита информации. Управление компьютерными инцидентами. Общие положения. — М., 2022.
3. ГОСТ Р 59711–2022. Защита информации. Управление компьютерными инцидентами. Организация деятельности по управлению компьютерными инцидентами. — М., 2022.
4. ГОСТ Р 59712–2022. Защита информации. Управление компьютерными инцидентами. Руководство по реагированию на компьютерные инциденты. — М., 2022. — 20 с.
5. Милославская Н. Г., Сенаторов М. Ю., Толстой А. И. Управление инцидентами информационной безопасности и непрерывностью бизнеса: учебное пособие для вузов. — 2-е изд., испр. — М.: Горячая линия — Телеком, 2014. — 170 с.
6. Милославская Н. Г., Сенаторов М. Ю., Толстой А. И. Проверка и оценка деятельности по управлению информационной безопасностью: учебное пособие для вузов. — М.: Горячая линия — Телеком, 2014. — 145 с.

# Методы распознавания мостов с летательных аппаратов

Саитов Марат Айратович, студент
Научный руководитель: Егорчев Антон Александрович, кандидат технических наук, доцент
Казанский (Приволжский) федеральный университет

*В статье рассматривается задача автоматического распознавания и сегментации мостов на снимках с беспилотных летательных аппаратов. Проведен сравнительный анализ двух моделей: YOLOv26-seg и Grounded-SAM2. Описаны особенности обеих моделей, причины их различия в точности и скорости. Экспериментально оценены метрики точности (mAP@0.5, mAP@0.5:0.95) и скорости инференса (FPS). Показано, что выбор модели зависит от сценария применения: YOLOv26-seg предпочтителен для real-time обработки, тогда как Grounded-SAM2 обеспечивает высокую точность без нужды в дообучении, что целесообразно для разметки данных.*

***Ключевые слова:*** *распознавание объектов, экземплярная сегментация, YOLO, SAM2, foundation-модели, мосты.*

## Введение

Для задачи распознавания объектов с летательных аппаратов характерна большая специфика данных, где объекты на изображении могут иметь маленький размер, также при движении БПЛА происходит резкое изменение фона, позиции объекта и возможны появления шумов. Учитывая такие ограничения, использование классических алгоритмов становится невозможным в связи их неспособности хорошо работать в меняющихся условиях [1].

Современные модели можно разделить на две категории:

1. Обучаемые модели (YOLO [4], R-CNN [5], …) — модели, обеспечивающие высокую скорость, но сильно зависят от количества и качества данных, на которых они обучались.

2. Foundation модели (SAM [6], Grounded-SAM [8]) — предобученные модели, способные распознавать объекты без дополнительного обучения, но требуют на входе промпт — текст, ограничивающий регион объекта, точка центра объекта, в зависимости от модели. Имеют хорошую точность, но требуют высокую вычислительную мощность.

В статье [2] проведен бенчмарк моделей YOLOv5/v8, где авторы выявили модели, обеспечивающие оптимальный баланс точности (лучший $\mathrm{mAP} \approx 0.88$), скорости инференса ($\mathrm{FPS} \approx 40$) и сформулировали рекомендации по развертыванию моделей на бортовых устройствах. Это подтверждает, что модели семейства YOLO являются хорошим вариантом для real-time систем. В то же время, исследования применения foundation моделей к аэрофотоснимкам демонстрирует их потенциал в сценариях с ограниченной разметкой. В статье [3] авторы оценили производительность моделей SAM [6] и SAM2 [7] для сегментации трещин на ледниках со снимков с дронов, показав результат $\mathrm{IoU} \approx 0.28$, что указывает на необходимость в дообучении либо в наличии другой модели, как Grounding DINO, которая будет давать моделям SAM промпты с ограничивающими рамками, что теоретически позволит лучше выделять объекты.

Цель данной статьи — провести сравнительный анализ архитектур YOLOv26-seg и Grounded-SAM2 для задачи распознавания мостов на снимках с БПЛА и сформулировать практические рекомендации по выбору модели в зависимости от эксплуатационных ограничений.

### Описание методов

Для поставленной задачи распознавания мостов с летательных аппаратов рекомендуется использовать обучаемые модели, учитывая, что важна real-time скорость, но из-за ограниченного количества данных рассмотрится еще и foundation модель. Среди множества архитектур распознавания объектов выделяются модели:

— YOLOv26-seg [9] — state-of-the-art решение среди семейства YOLO для real-time сегментации. Модель является одноэтапной с единой базовой моделью, включающей C3k2, C2PSA блоки и раздельными головами для детекции и генерации масок. Ключевыми особенностями является anchor-free подход, отсутствие NMS на инференсе и Dual Label Assignment для устойчивости к малым объектам.

— Grounded-SAM2 является предобученной моделью, которая состоит из двух моделей — Grounding DINO и SAM2, где каждая из них отдельно обучена на 10 миллионах наборах изображений и нужных для них промптах. Способна распознавать всевозможные классы учитывая текстовое описание, приложенное пользователем. Имеет хорошую точность, но из-за сложной структуры, содержащей две модели и механизмы внимания, требует больших вычислительных мощностей.

### Описание датасета

Для выбора подходящей модели будет проведен сравнительный экспериментальный анализ, где сравнятся метрики точности mAP и скорости инференса FPS. Эксперимент будет проводиться на собранном датасете с мостами, который размечен, изначально используя модель Grounded-SAM2 и в дальнейшем подкорректирован вручную, что сэкономило время разметки.

Датасет состоит из ~1500 размеченных изображений мостов масками в формате YOLO, который разделен пропорциями 75/15/10 на тренировочный, валидационный и тестовый выборки соответственно. Пример изображений с датасета изображен на рис. 1.


Рис. 1. **Пример данных с датасета**

### Обучение

Обучение производилось через библиотеку Ultralytics используя гиперпараметры указанные в таблице 1.

Таблица 1. **Гиперпараметры обучения модели**

| Параметр | Значение |
|---|---|
| Количество эпох | 300 |
| Размер батча | 32 |
| Размер входных изображений | 640x640 |
| Оптимизатор | AdamW |

Также при обучении добавлялись аугментации, указанные в таблице 2.

Таблица 2. **Аугментации, применяемые во время обучения**

| Аугментация | Значение |
|---|---|
| Поворот изображения (degrees) | 5° |
| Сдвиг изображения (translate) | 0.005 (5 %) |
| Масштабирование (scale) | 0.2 (20 %) |
| Горизонтальное отражение (fliplr) | 0.5 (50 % вероятность горизонтального отражения) |
| Изменение оттенка (hsv_h) | 0.015 |
| Изменение насыщенности (hsv_s) | 0.3 |
| Изменение яркости (hsv_v) | 0.3 |

**Результаты**

По результатам Grounded-SAM2 демонстрирует преимущество по точности сегментации (+6.5 % по mAP@0.5, +22.6 % по mAP@0.5:0.95), что объясняется предобучением на масштабных данных и способности к zero-shot обобщению. YOLOv26-seg превосходит по скорости инференса (в 58.9 раз выше FPS), что делает её пригодной для real-time систем. В таблице 3 указаны полученные метрики моделей.

Таблица 3. **Результаты моделей**

| Модель | mAP 50 | mAP50–95 | FPS |
|---|---|---|---|
| YOLO26n-seg | 0.932 | 0.517 | 91.23 |
| Grounded-SAM2 | 0.993 | 0.634 | 1.55 |

Замеры FPS обеих моделей выполнены на графической видеокарте NVIDIA Quadro RTX 8000 (48 GB GDDR6) в режиме инференса без учета пред/постобработки.

Визуальный анализ на рис. 2 показывает, что различия в качестве масок минимальны для крупных объектов.


Рис. 2. **Маска до и после инференса моделей**

**Выводы**

На основе проделанного анализа YOLOv26-seg обеспечивает более высокую производительность для real-time систем с ограничением по времени отклика, что подтверждается высоким FPS (~91 FPS). Grounded-SAM2 имеет лучшую

точность (mAP@0.5:0.95 = 0.634, +22.6 % относительно точности YOLO) и работает без нужды в дообучении, что является хорошим решением для автоматизации процесса аннотации изображений для дальнейшего обучения моделей.

Литература:


1. Liu Y. A survey of small object detection based on deep learning in aerial images [Электронный ресурс] / Y. Liu, X. Zhang, Z. Wang // Artificial Intelligence Review. — 2025. — URL: https://link.springer.com/article/10.1007/s10462–025–11150–9 (дата обращения: 03.03.2026).
2. Phan T.-N. Deep Learning Models for UAV-Assisted Bridge Inspection: A YOLO Benchmark Analysis [Электронный ресурс] / T.-N. Phan, H.-H. Nguyen, T.-T.-H. Ha, H.-T. Thai, K.-H. Le // 2024 International Conference on Advanced Technologies for Communications (ATC). — Ho Chi Minh City, Vietnam, 17–19 Oct. 2024. — DOI: 10.1109/ATC63255.2024.10908331. — URL: https://ieeexplore.ieee.org/document/10908331 (дата обращения: 03.03.2026).
3. Wallace S. Exploring Segment Anything Foundation Models for Out of Domain Crevasse Drone Image Segmentation [Электронный ресурс] / S. Wallace, A. Durrant, W. D. Harcourt, R. Hann, G. Leontidis // Proceedings of the Neural Information Processing Systems Workshop on Deep Learning for Climate Change and Sustainability (NLDL 2025). — 2024. — URL: https://openreview.net/forum?id=CGkyjTXomz (дата обращения: 04.03.2026).
4. Redmon J. You Only Look Once: Unified, Real-Time Object Detection [Электронный ресурс] / J. Redmon, S. Divvala, R. Girshick, A. Farhadi // arXiv. — 2015. — № 1506.02640. — URL: https://arxiv.org/abs/1506.02640 (дата обращения: 12.03.2026).
5. Girshick R. Rich feature hierarchies for accurate object detection and semantic segmentation [Электронный ресурс] / R. Girshick, J. Donahue, T. Darrell, J. Malik // arXiv. — 2013. — № 1311.2524. — URL: https://arxiv.org/abs/1311.2524 (дата обращения: 06.03.2026).
6. Segment Anything [Электронный ресурс] // arXiv preprint. — 2023. — № 2304.02643. — URL: https://arxiv.org/abs/2304.02643 (дата обращения: 25.03.2026).
7. SAM 2: Segment Anything in Images and Videos [Электронный ресурс] // arXiv preprint. — 2024. — № 2408.00714. — URL: https://arxiv.org/abs/2408.00714 (дата обращения: 26.03.2026).
8. Grounded SAM: Assembling Open-World Models for Diverse Visual Tasks [Электронный ресурс] // arXiv preprint. — 2024. — № 2401.14159. — URL: https://arxiv.org/abs/2401.14159 (дата обращения: 27.03.2026).
9. Ultralytics YOLO26: Key Architectural Enhancements and Performance Improvements [Электронный ресурс] // arXiv. — 2025. — № 2509.25164. — URL: https://arxiv.org/abs/2509.25164 (дата обращения: 20.03.2026).


## Архитектура экосистемы и UX-паттерны суперприложений Китая и России

Смирнова Алёна Николаевна, студент
МИРЭА — Российский технологический университет (г. Москва)

*Цель. Определить, как степень интеграции сервисов влияет на маршруты входа, границы контекста, плотность главного экрана и роль QR-кода. Метод. Проведен встроенный сравнительный анализ шести кейсов с функциональным спариванием: Weixin/WeChat и VK, Alipay и СберБанк Онлайн, Meituan и Яндекс Go. Для каждого кейса заполнена кодировочная карта из семи показателей. Шкалы и правила фиксации заданы до межкейсового сопоставления. Материал включает официальные изображения интерфейсов, документацию, карточки приложений, отчеты компаний и научные публикации по состоянию на 24 июля 2026 г. Результаты. Все три китайских кейса получили оценку 2 по интеграции runtime. Среди российских продуктов такая оценка установлена для Яндекс Go в границах городских услуг; VK и СберБанк Онлайн имеют смешанный профиль. Число документированных маршрутов входа составляет 3–5 для китайских платформ и 3 для российских. QR-код выполняет системную функцию в WeChat и Alipay, тогда как в российских кейсах он обслуживает отдельные операции. Выводы. Наблюдаемые различия связаны с архитектурой runtime и общими платежно-идентификационными примитивами. Культурные факторы сохраняют статус проверяемой гипотезы и требуют пользовательского исследования.*

***Ключевые слова:*** *суперприложение, пользовательский опыт, UX-паттерны, цифровая экосистема, мини-программы, WeChat, Alipay, VK, СберБанк Онлайн, Яндекс Go.*

### Введение

Суперприложение объединяет разнородные услуги в одной пользовательской оболочке и предоставляет разработчикам общие платформенные ресурсы: учетную запись, платежный контур, поиск, разрешения и среду исполнения ми-

ни-программ [1–5]. Его интерфейс отражает устройство платформы. Сетка сервисов, глубина переходов и способы возврата зависят от того, где проходят технические и организационные границы.

Китайские платформы развили модель приложения внутри приложения. WeChat связывает коммуникацию с Weixin Pay и мини-программами; Alipay расширяет кошелек до платформы цифровых услуг; Meituan концентрирует локальную коммерцию и городские операции [16–20]. Российские компании чаще сохраняют самостоятельные клиенты. VK, Сбер и Яндекс используют единый ID и общие платежные механизмы, однако глубина встраивания сервисов различается [21–26].

Популярная формула о «плотных китайских» и «минималистичных российских» интерфейсах описывает внешний результат. Причинное объяснение требует проверки архитектуры runtime, способов входа и платежной инфраструктуры. Исследования подробно рассматривают платформизацию WeChat и мини-программы [2, 3, 15], тогда как сопоставление с российскими продуктами на уровне операционализированных UX-признаков встречается реже.

Работа отвечает на три вопроса. RQ1 устанавливает связь архитектуры с плотностью и границами навигации. RQ2 рассматривает различия в обнаружении сервисов, авторизации, QR-сценариях и оплате. RQ3 проверяет пределы культурного объяснения после учета технических и регуляторных факторов.

Вклад исследования состоит в прозрачной кодировочной схеме и в ее применении к шести продуктам. Результат представляет профиль архитектурных компромиссов. Он не служит рейтингом приложений и не заменяет измерение пользовательской эффективности.

## 1. Теоретические основания

### *1.1. Суперприложение как вложенная платформа*

Архитектура цифровой платформы включает ядро, интерфейсы доступа и дополняющие модули [6, 7]. Для супераппа в состав ядра входят пользовательские примитивы: ID, платежи, разрешения, уведомления и механизмы распространения сервисов. Мини-программа исполняется по правилам принимающей платформы и наследует часть ее контекста.

На одном полюсе находится единая оболочка со встроенным runtime. На другом полюсе расположена федеративная экосистема самостоятельных приложений, связанных SSO, deep links и программами лояльности. Гибридная конфигурация встраивает частые короткие действия и оставляет специализированные операции в отдельных клиентах. Маркетинговое название продукта не определяет его позицию на этом континууме.

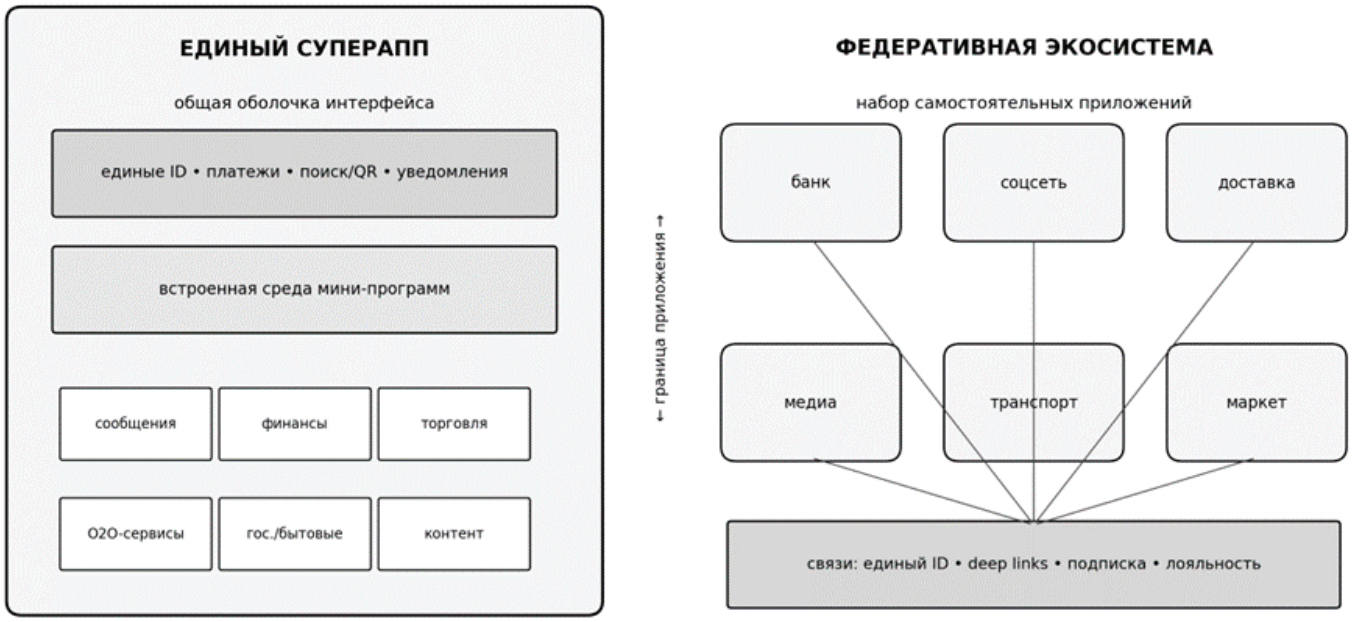


Рис. 1. **Предельные модели архитектуры цифровой экосистемы (составлено автором)**

WeChat снижает издержки установки: сервис открывается из поиска, сообщения, публичного аккаунта или QR-кода и работает внутри Weixin [2, 3]. Данная модель ускоряет переход от обнаружения к действию. Одновременно владелец платформы контролирует правила публикации, доступ к данным и часть платежного маршрута.

*1.2. Архитектурные предпосылки UX-паттернов*

UX-паттерн в этой работе означает воспроизводимый способ организации действия: вход через сервисный хаб, запуск по QR-коду, сохранение сессии или возврат в общую оболочку. Видимый компонент выступает наблюдаемым следствием системной возможности.

Плотность растет, когда десятки функций должны оставаться обнаружимыми в одной точке входа. Интерфейс отвечает сетками иконок, поиском, вкладками и персонализированными блоками. Федеративная модель распределяет нагрузку между приложениями. Локальный экран становится проще, но сквозная задача включает переходы через операционную систему или магазин приложений.

Глубину следует разделять на внутрисервисную и сквозную. Специализированный клиент часто сокращает путь к своей основной функции. Пользовательский маршрут между продуктами при этом удлиняется из-за смены контекста. Суперапп способен быстро открыть редкий сервис через поиск или QR, хотя выбор внутри сервисного хаба остается отдельной нагрузкой.

Платежи и идентификация меняют финальную часть любого сценария. Общий кошелек позволяет завершить действие без повторного ввода реквизитов. Межбанковская инфраструктура требует явного выбора приложения и способа оплаты. В 2025 г. через СБП проведено 18,3 млрд операций, а оплату принимали 3 млн торговых предприятий [27]. Массовый QR в России поддерживает интероперабельность без обязательной привязки к одному коммерческому супераппу.


Рис. 2. **Аналитическая цепочка от архитектуры к UX-эффектам (составлено автором)**

*1.3. Культурная рамка*

Кросс-культурные модели связывают представление информации с отношением к иерархии, неопределенности и контексту [11–13]. Они полезны для постановки гипотез о насыщенности экрана и роли социальных рекомендаций. Национальные индексы описывают агрегаты и не характеризуют отдельного пользователя [14]. Поэтому культура рассматривается после архитектурных и инфраструктурных факторов.

Повседневная встроенность WeChat формировалась вместе с распространением платежей, QR-кодов и мини-программ [3, 15]. Пользовательская привычка усиливает платформенный паттерн, а доступность сканера расширяет число физических точек входа. Разделение этих эффектов требует данных о поведении пользователей.

**2. Методология**

*2.1. Дизайн и выбор кейсов*

Исследование имеет встроенный сравнительный многокейсовый дизайн. Каждый продукт рассматривается как самостоятельный кейс, а повторяющиеся элементы пользовательского маршрута служат вложенными единицами наблюдения. Многокейсовость здесь означает последовательное применение одной кодировочной карты к шести случаям и поиск повторяющихся либо контрастных конфигураций. Логика соответствует аналитической репликации: вывод поддерживается, когда один механизм воспроизводится в нескольких кейсах [8, с. 39–42; 29, с. 25–32].

Выборка целевая. Внутри каждой пары продукты имеют сходное функциональное ядро, но различаются глубиной интеграции. WeChat и VK представляют коммуникационные платформы с мини-приложениями. Alipay и СберБанк Онлайн начинают маршрут из финансового контекста. Meituan и Яндекс Go связывают цифровой заказ с физической городской услугой. Такое спаривание уменьшает влияние предметной области и сохраняет архитектурный контраст.

Дизайн относится к формату коротких множественных кейсов: на каждый продукт приходится ограниченный, но однородный набор открытых источников. Для данного формата критичны явные границы кейса, единый протокол и прослеживаемость утверждений [30, с. 357–363]. Граница кейса установлена как публичная мобильная оболочка и документированные платформенные возможности на 24 июля 2026 г.

*2.2. Материал и единицы наблюдения*

Для каждого продукта использованы официальная карточка приложения либо интерфейсный пресс-материал, описание архитектуры или документация, публикации компании о составе сервисов и академический источник для интерпретации. Функциональное утверждение включалось в кодировочную карту после подтверждения двумя типами источников. Внешний вид экрана фиксировался по официальному изображению с датой обращения.

Наблюдение охватывало три этапа маршрута. Первый этап фиксировал обнаружение и вход в сервис. Второй этап проверял сохранение идентификации и платежного контекста. Третий этап описывал возврат, статус или историю операции. Время выполнения и число нажатий не измерялись: открытые изображения не дают достаточной основы для таких показателей.

*2.3. Кодировочная схема и метрики*

Кодировочная карта содержит семь показателей. Четыре описывают платформенную архитектуру, три фиксируют интерфейсное проявление. Для порядковых шкал заданы якорные значения 0, 1 и 2. Показатель CP является счетчиком общих примитивов, ER отражает число документированных маршрутов входа. Суммарный балл не рассчитывался, поскольку показатели выражают разные свойства.

Таблица 1. **Операционализация показателей**

| Код | Показатель | Правило фиксации | Источник данных |
|---|---|---|---|
| RI | Интеграция runtime | 0: отдельный клиент; 1: смешанная модель; 2: встроенное исполнение | Документация платформы, карточка приложения |
| CP | Общие примитивы | 0–4: по одному баллу за ID, платеж, статус/уведомление и сохранение контекста | Документация, описание пользовательского потока |
| ER | Маршруты входа | 0–5: хаб, поиск, QR, сообщение/deep link, контекстная рекомендация | Официальные интерфейсные материалы |
| CX | Граница контекста | 0: одна оболочка; 1: отдельный раздел; 2: внешний клиент или браузер | Описание маршрута, интерфейс |
| HD | Плотность хаба | 0: до 6 входов; 1: 7–12; 2: более 12 в первом экране | Официальный скриншот |
| QR | Центральность QR | 0: периферийно; 1: отдельная операция; 2: обнаружение и выполнение сервиса | Документация, интерфейс |
| UC | Непрерывность UX | 0: общей рамки нет; 1: частичная; 2: стабильная оболочка и возврат | Интерфейс, правила платформы |

Первичное кодирование дало 42 ячейки: шесть кейсов по семи показателям. Затем каждая ячейка повторно сверялась с источником; неоднозначные значения записывались диапазоном. Эвристики Нильсена использовались для интерпретации видимости состояния, контроля и согласованности [9, 10]. Они не входили в числовой профиль.

*2.4. Достоверность и границы вывода*

Конструктная валидность поддерживается операциональными определениями и одинаковыми источниковыми требованиями. Внутренняя валидность ограничена наблюдательным дизайном, поэтому причинные формулировки имеют статус объяснительного механизма. Аналитическая переносимость проверяется повторением профиля внутри функциональных пар. Статистическое обобщение на все приложения Китая и России не выполняется.

Кодирование проведено одним исследователем. Межэкспертное согласие не рассчитывалось. Повторная сверка уменьшает риск фактической ошибки, но не устраняет субъективность при выборе границы между соседними категориями.

## 3. Результаты

### *3.1. Архитектурный профиль*

Три китайских платформы имеют встроенное исполнение в исследуемом домене. WeChat и Alipay допускают сторонние мини-программы, Meituan объединяет поставщиков локальной коммерции в общей транзакционной среде. В российской группе Яндекс Go достигает сопоставимой интеграции внутри городских услуг. VK и СберБанк Онлайн совмещают встроенные функции с отдельными клиентами.

Таблица 2. **Архитектурные показатели кейсов**

| Платформа | RI | CP | ER | CX |
|---|---|---|---|---|
| WeChat | 2 | 4 | 5 | 0–1 |
| Alipay | 2 | 4 | 4 | 0–1 |
| Meituan | 2* | 4 | 3 | 0–1 |
| VK | 1 | 3 | 3 | 0–2 |
| СберБанк Онлайн | 1 | 4 | 3 | 0–2 |
| Яндекс Go | 2* | 4 | 3 | 0–1 |

* Оценка 2 относится к домену платформы: локальная коммерция для Meituan и городские услуги для Яндекс Go. ER показывает число подтвержденных маршрутов, поэтому значение 5 у WeChat связано с поиском, QR, хабом, сообщениями и контекстными ссылками [2, 3, 16]. Диапазон CX фиксирует смешанные маршруты.

Таблица 3. **Интерфейсные показатели кейсов**

| Платформа | HD | QR | UC |
|---|---|---|---|
| WeChat | 1 | 2 | 2 |
| Alipay | 2 | 2 | 2 |
| Meituan | 2 | 1 | 2 |
| VK | 1 | 0 | 1 |
| СберБанк Онлайн | 1 | 1 | 1 |
| Яндекс Go | 1 | 0 | 2 |

Высокая плотность Alipay и Meituan соответствует широкому рынку услуг. WeChat сохраняет умеренную плотность коммуникационного ядра и переносит часть выбора в поиск и мини-программы. Российские продукты чаще разделяют домены по вкладкам или приложениям. Непрерывность Яндекс Go оценивается в 2 в пределах общей оболочки городских сервисов.

### *3.2. WeChat и VK: коммуникационное ядро*

Обе платформы начинают с общения. Скриншоты официальных карточек показывают сходный базовый объект: список чатов и экран сообщения (рис. 3). Дальнейшая архитектура расходится. WeChat открывает мини-программы внутри стабильной оболочки, сохраняя ID и доступ к Weixin Pay. VK Mini Apps используют VK ID и платформенные API, однако соседние продукты VK сохраняют собственные клиенты [25, 26].

Пять маршрутов входа WeChat уменьшают зависимость от запоминания положения сервиса. Физический QR-код может сразу открыть цифровое действие. В VK переход к мини-приложению также возможен из социальной среды, но часть сквозных задач выводит пользователя в отдельный продукт. Значение CX поэтому изменяется от 0 до 2.

Пара отвечает на RQ1 через место концентрации сложности. WeChat переносит ее в сервисный слой общей платформы. VK распределяет функции между социальной оболочкой и самостоятельными приложениями. Для RQ2 различие фиксируется показателем QR: системное значение 2 у WeChat и периферийное значение 0 у VK.

### *3.3. Alipay и СберБанк Онлайн: расширение финансового ядра*

Финансовая учетная запись создает основу для расширения в транспорт и повседневные услуги. Alipay размещает сканирование, перевод, поездки и мини-программы в одном сервисном хабе. Текущий интерфейс показывает крупный

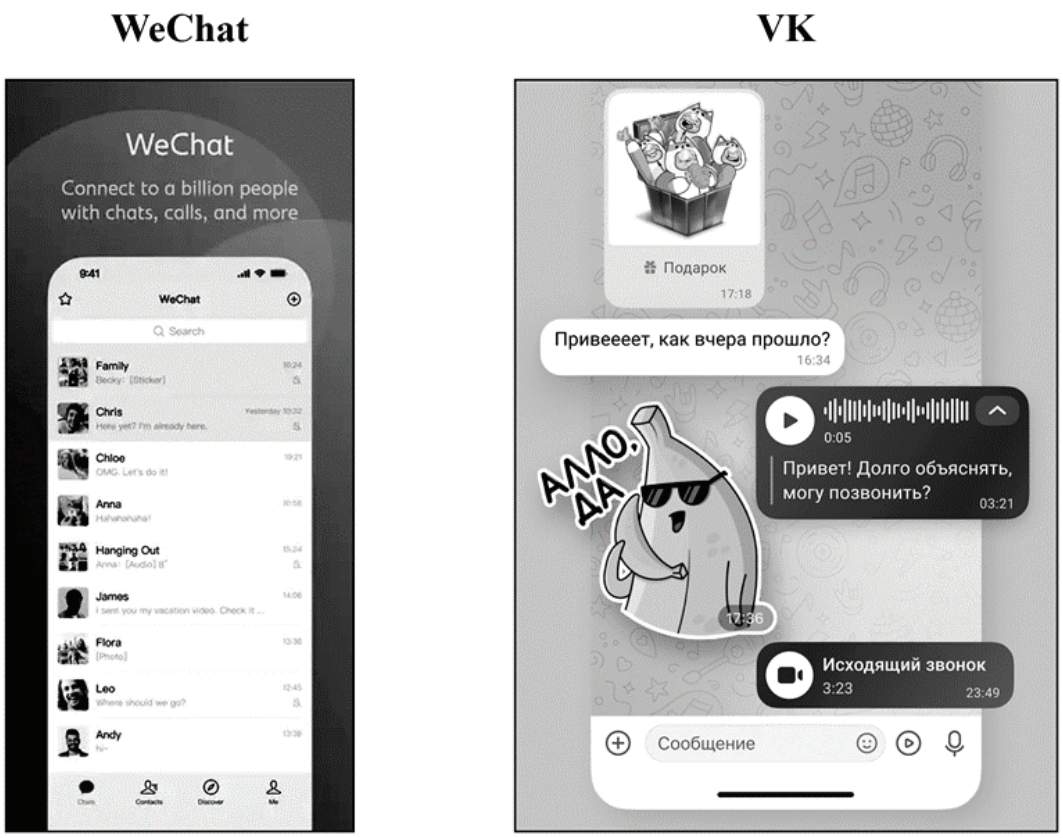


Рис. 3. **Коммуникационные экраны WeChat и VK (источники изображений: [31; 32], адаптировано автором)**

набор входов уже на первом экране (рис. 4). СберБанк Онлайн отделяет вкладку «Для жизни» от платежей и кредитов, сохраняя банковскую структуру нижней навигации [21, 22].


Рис. 4. **Сервисные хабы Alipay и СберБанк Онлайн (источники изображений: [33; 22], адаптировано автором)**

Alipay получил HD=2 и QR=2. Сканер работает как вход в платеж и услугу, а кошелек остается частью того же потока [18, 19]. СберБанк Онлайн получил HD=1 и QR=1. Общий ID и платежный инструмент снижают повторный ввод данных, однако пользователь может перейти в отдельный продукт экосистемы.

Российская СБП усиливает альтернативный маршрут. Универсальный QR позволяет выбрать банковское приложение [27, 28]. Интерфейс должен явно показывать платежный контекст и получателя. Зависимость от одного супераппа при этом снижается.

*3.4. Meituan и Яндекс Go: городской O2O*

Обе платформы связывают экран с физическим местом и исполнителем. Meituan охватывает рестораны, гостиницы, развлечения, доставку и торговлю [20]. Каталожная широта отражена в множестве иконок, карточек и предложений. Яндекс Go организует главный экран вокруг режимов городской задачи и общей карты (рис. 5).

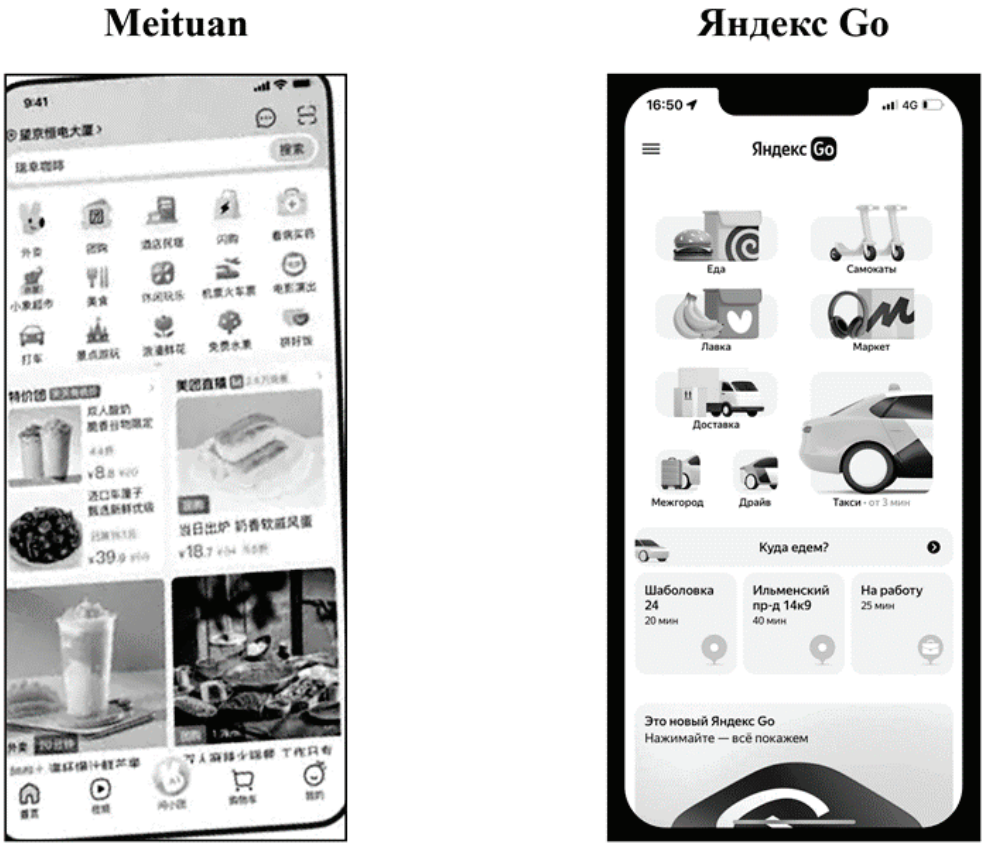


Рис. 5. **Главные экраны Meituan и Яндекс Go (источники изображений: [34; 24], адаптировано автором)**

Meituan получил HD=2: каталог должен поддерживать поиск поставщика, сравнение и повтор заказа. Яндекс Go получил HD=1 и UC=2. Крупные входы ведут в отдельные режимы, а активные заказы возвращаются в общую оболочку [24]. Карта, адрес и платежный профиль служат общими объектами.

В декабре 2025 г. двумя и более городскими сервисами Яндекса пользовались 19 % активной аудитории; доля оборота направлений вне такси превысила 49 % [23]. Эти данные подтверждают доменную централизацию. Граница суперaппа совпадает с городскими задачами, а медиапродукты Яндекса остаются за ее пределами.

*3.5. Межкейсовое сопоставление*

Профили поддерживают связь между RI и UC. Четыре кейса с RI=2 получили UC=2 в обозначенном домене. В VK и СберБанк Онлайн смешанный runtime сопровождается диапазоном CX до 2 и частичной непрерывностью. Вывод основан на шести наблюдениях и имеет аналитический характер.

Связь плотности с архитектурой менее линейна. WeChat сочетает RI=2 с HD=1, поскольку коммуникационное ядро отделено от сервисного слоя. Alipay и Meituan показывают RI=2 и HD=2. Плотность зависит от широты каталога и выбранной точки агрегации.

## 4. Обсуждение

### *4.1. Ответ на RQ1: место концентрации сложности*

Единая оболочка переносит сложность из межприложенных переходов в обнаружение сервисов. Поиск, недавние действия и контекстные входы компенсируют широкий каталог. Федеративная модель дает специализированному продукту короткий локальный маршрут, но увеличивает число границ в сквозной задаче. Оценивать плотность следует вместе с функцией экрана и показателем CX.

Сводный результат уточняет исходное предположение. Архитектура объясняет положение сложности, а широта рынка определяет ее визуальную величину. Минималистичное коммуникационное ядро WeChat и плотный Alipay существуют внутри одной национальной среды.

### *4.2. Ответ на RQ2: QR и общий платежный слой*

В WeChat и Alipay QR-код адресует сервис внутри приложения с готовой идентичностью. Пользователь пропускает поиск отдельного клиента. В СБП QR адресует платеж, а выбор приложения сохраняется за пользователем. Две модели могут давать короткий маршрут, однако распределяют контроль между участниками по-разному.

Единая авторизация уменьшает трение и повышает цену ошибки. Встроенный сервис должен показывать поставщика, область разрешений и способ отзыва доступа. Платформенная согласованность не должна скрывать юридическую границу.

### *4.3. Ответ на RQ3: предел культурного объяснения*

Культурная гипотеза остается возможной после учета архитектуры, опыта и регулирования. Представленная выборка не содержит пользовательских данных, поэтому связь национальных характеристик с предпочтением плотности не проверена. Для такой проверки нужны сопоставимые задания и группы участников из двух стран.

## 5. Рекомендации для проектирования

Проектирование следует начинать с перечня общих примитивов и границы runtime. Главный экран строится после решения о том, какие действия сохраняют сессию и платежный контекст. Механическое добавление иконок увеличивает HD без роста UC.

Для единой оболочки нужны стабильный возврат, журнал разрешений и объяснение поставщика услуги. Для федеративной модели критичны deep links в конкретный шаг, сохранение состояния и единая терминология. Гибридная модель должна зафиксировать основной маршрут для каждой задачи и устранить дублирование.

Продуктовая аналитика должна разделять локальную и сквозную эффективность. Минимальный набор включает успешность задачи, время до первого осмысленного действия, число контекстных границ и долю повторной авторизации. Эти данные позволят проверить выводы настоящего кабинетного исследования.

## 6. Ограничения

Выборка ограничена шестью крупными платформами. Персонализация, регион и версия операционной системы меняют расположение элементов. Официальные карточки приложений показывают отобранные экраны и могут иметь рекламную композицию. Поэтому HD трактуется как признак опубликованного интерфейса.

В работе отсутствуют логи, айтрекинг и контролируемые тесты. Порядковые значения описывают архитектуру и интерфейсные возможности. Они не являются измерением удобства. Временной срез завершен 24 июля 2026 г.; обновление продукта может изменить отдельную ячейку профиля.

## Заключение

Сравнительный дизайн показал устойчивое различие между встроенным и смешанным runtime. В китайской группе RI=2 зафиксирован у WeChat, Alipay и Meituan. В российской группе такое значение получил Яндекс Go в домене городских услуг. VK и СберБанк Онлайн сохранили переходы к самостоятельным продуктам.

RQ1 получает следующий ответ: архитектура определяет место навигационной сложности. RQ2 связывает системную роль QR с общей идентичностью и платежным слоем. По RQ3 данных для культурного вывода недостаточно; национальная культура остается гипотезой для пользовательского эксперимента.

Проверяемым итогом служит матрица из 42 кодировочных ячеек. В четырех кейсах с RI=2 непрерывность UC также равна 2 в заявленном домене; в двух кейсах с RI=1 значение UC равно 1.

Литература:


1. Hasselwander M. Digital platforms' growth strategies and the rise of super apps // Heliyon. 2024. Vol. 10, no. 5. Art. e25856. DOI: 10.1016/j.heliyon.2024.e25856.
2. Cheng K., Schreieck M., Wiesche M., Krcmar H. Emergence of a Post-App Era: An Exploratory Case Study of the WeChat Mini-Program Ecosystem // Wirtschaftsinformatik 2020 Proceedings. 2020. P. 1444–1458. DOI: 10.30844/wi_2020_n1-cheng.
3. Jia L., Nieborg D. B., Poell T. On super apps and app stores: digital media logics in China's app economy // Media, Culture & Society. 2022. Vol. 44, no. 8. P. 1437–1453. DOI: 10.1177/01634437221128937.
4. Steinberg M. LINE as Super App: Platformization in East Asia // Social Media + Society. 2020. Vol. 6, no. 2. DOI: 10.1177/2056305120933285.
5. Yu M. The Rise of Super Apps: Insights and Opportunities from a UX Perspective // 2023 IEEE International Professional Communication Conference. 2023. P. 128–133.
6. Tiwana A. Platform Ecosystems: Aligning Architecture, Governance, and Strategy. Waltham: Morgan Kaufmann, 2014. 300 p.
7. de Reuver M., Sørensen C., Basole R. C. The Digital Platform: A Research Agenda // Journal of Information Technology. 2018. Vol. 33, no. 2. P. 124–135. DOI: 10.1057/s41265–016–0033–3.
8. Yin R. K. Case Study Research and Applications: Design and Methods. 6th ed. Thousand Oaks: SAGE Publications, 2018. 352 p.
9. Nielsen J., Molich R. Heuristic evaluation of user interfaces // Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. 1990. P. 249–256. DOI: 10.1145/97243.97281.
10. Nielsen J. Heuristic Evaluation // Usability Inspection Methods / ed. by J. Nielsen, R. L. Mack. New York: John Wiley & Sons, 1994. P. 25–62.
11. Marcus A., Gould E. W. Crosscurrents: cultural dimensions and global Web user-interface design // Interactions. 2000. Vol. 7, no. 4. P. 32–46. DOI: 10.1145/345190.345238.
12. Hofstede G. Dimensionalizing Cultures: The Hofstede Model in Context // Online Readings in Psychology and Culture. 2011. Vol. 2, no. 1. DOI: 10.9707/2307–0919.1014.
13. Hall E. T. Beyond Culture. New York: Anchor Books, 1976. 320 p.
14. Venaik S., Brewer P. Critical issues in the Hofstede and GLOBE national culture models // International Marketing Review. 2013. Vol. 30, no. 5. P. 469–482. DOI: 10.1108/IMR-03–2013–0058.
15. Huang Y., Miao W. Re-domesticating social media when it becomes disruptive: Evidence from China's "super-sticky" WeChat and beyond // Mobile Media & Communication. 2021. Vol. 9, no. 2. P. 177–194. DOI: 10.1177/2050157920940765.
16. Tencent. Weixin/WeChat: A Way of Life [Электронный ресурс]. URL: https://www.tencent.com/products/weixin-wechat/ (дата обращения: 24.07.2026).
17. Tencent. Uber Launches Weixin Mini Program to Unlock Seamless Global Travel for Chinese Users [Электронный ресурс]. 2025. URL: https://www.tencent.com/en-us/articles/2202137.html (дата обращения: 24.07.2026).
18. Ant Group. Digital Payment [Электронный ресурс]. URL: https://www.antgroup.com/en/business-development (дата обращения: 24.07.2026).
19. Ant Group. Alipay Records Surge of China Inbound and Outbound Travel Spending During the 2025 Chinese New Year [Электронный ресурс]. 2025. URL: https://www.antgroup.com/en/news-media/press-releases/1738735200000 (дата обращения: 24.07.2026).
20. Meituan. About Us: A Tech-driven Retail Company [Электронный ресурс]. URL: https://www.meituan.com/en-US/about-us (дата обращения: 24.07.2026).
21. Сбер. Команда СберБанк Онлайн [Электронный ресурс]. URL: https://developers.sber.ru/kak-v-sbere/teams/sberbank-online (дата обращения: 24.07.2026).
22. СберБанк Онлайн: описание приложения в RuStore [Электронный ресурс]. URL: https://www.rustore.ru/catalog/app/ru.sberbankmobile (дата обращения: 24.07.2026).
23. Яндекс. Финансовые результаты за IV квартал 2025 года и 2025 год [Электронный ресурс]. 2026. URL: https://yandex.ru/company/news/17–02–2026 (дата обращения: 24.07.2026).
24. Яндекс. Яндекс Go обновил главный экран [Электронный ресурс]. 2023. URL: https://yandex.ru/company/news/04–07–23 (дата обращения: 24.07.2026).
25. VK. Что такое VK [Электронный ресурс]. URL: https://vk.company.ru/ru/company/about/ (дата обращения: 24.07.2026).
26. VK. Пресс-релиз по результатам за I квартал 2025 года [Электронный ресурс]. 2025. URL: https://vk.company.ru/ru/investors/info/12005/ (дата обращения: 24.07.2026).

27. Банк России. СБП: основные показатели за IV квартал 2025 года [Электронный ресурс]. 2026. URL: https://www.cbr.ru/analytics/nps/sbp/4_2025/ (дата обращения: 24.07.2026).
28. Банк России. Итоги работы Банка России 2025: развитие системы платежей и расчетов [Электронный ресурс]. 2026. URL: https://www.cbr.ru/about_br/publ/results_work/2025/razvitie-sistemy-platezhey-i-raschetov/ (дата обращения: 24.07.2026).
29. Eisenhardt K. M., Graebner M. E. Theory Building From Cases: Opportunities and Challenges // Academy of Management Journal. 2007. Vol. 50, no. 1. P. 25–32. DOI: 10.5465/amj.2007.24160888.
30. Käss S., Brosig C., Westner M., Strahringer S. Short and sweet: multiple mini case studies as a form of rigorous case study research // Information Systems and e-Business Management. 2024. Vol. 22. P. 351–384. DOI: 10.1007/s10257–024–00674–2.
31. WeChat: карточка приложения в App Store [Электронный ресурс]. URL: https://apps.apple.com/us/app/wechat/id414478124 (дата обращения: 24.07.2026).
32. ВКонтакте: чаты, видео, музыка: карточка приложения в RuStore [Электронный ресурс]. URL: https://www.rustore.ru/catalog/app/com.vkontakte.android (дата обращения: 24.07.2026).
33. Alipay: карточка приложения в App Store [Электронный ресурс]. URL: https://apps.apple.com/cn/app/id333206289 (дата обращения: 24.07.2026).
34. Meituan: карточка приложения в App Store [Электронный ресурс]. URL: https://apps.apple.com/cn/app/id423084029 (дата обращения: 24.07.2026).

# Программный модуль для контекстного анализа данных юридических документов

Тагунков Михаил Владиславович, студент
Национальный исследовательский университет «Московский институт электронной техники» (г. Зеленоград)

*В статье рассматривается программный модуль, предназначенный для преобразования неструктурированного текста юридических документов в набор структурированных фактов. Основное внимание уделено контекстному выделению дат событий, временных интервалов и даты рождения участника производства. Предлагаемый подход сочетает нормализацию текста, регулярные выражения, словари юридически значимых маркеров и эвристические правила разрешения неоднозначностей. Описаны архитектура модуля, этапы формирования кандидатов, анализ локального контекста, фильтрация служебных дат и представление результата с указанием происхождения каждого извлечённого факта. Модуль ориентирован на обработку судебных актов, приговоров, протоколов и архивных материалов.*

***Ключевые слова:*** *контекстный анализ, юридические документы, извлечение дат, обработка естественного языка, регулярные выражения, C#.*

## Введение

Юридические документы содержат значительный объём фактической информации, однако её форма редко является однородной. Одна и та же дата может обозначать момент совершения события, дату составления документа, время проведения экспертизы, дату вступления решения в силу либо дату рождения участника дела. При простом поиске календарных выражений все такие упоминания попадают в единый список, из-за чего результат требует длительной ручной проверки. Для прикладной обработки важен не сам факт наличия даты в тексте, а её смысловая роль и связь с описываемым событием. Поэтому задача извлечения хронологических сведений должна рассматриваться как задача контекстного анализа, включающая распознавание сущности, классификацию её назначения и сохранение связи с фрагментом исходного документа.

Обработка естественного языка включает нормализацию, поиск сущностей и их интерпретацию [1, с. 21–30]. Для юридического текста эти операции осложняются канцелярскими конструкциями, длинными предложениями, перечислениями эпизодов и повторным упоминанием фактов. Значение выражения часто определяется процессуальной ролью лица и видом документа [2]. Цель модуля состоит в выделении временных фактов и преобразовании их в структурированный результат: точные даты, интервалы неопределённости, дату рождения целевого лица, контекстные признаки и ссылку на исходный фрагмент.

## Постановка задачи контекстного анализа

Датой-кандидатом считается календарное выражение, которое может обозначать юридически значимый факт. Кандидаты включают полные числовые и словесные даты, упоминания месяца и года, а также конструкции «в начале марта 1991 года», «с марта по май 1991 года», «в период с 10 по 14 апреля». Сначала выражение рассматривается без предположения о назначении, затем анализируются

предложение, соседние фразы, заголовок раздела и ближайшие слова-маркеры. Найденная последовательность связывается с контекстом и приводится к нормализованному виду [3, с. 19–24].

Выделяются три категории: даты событий, персональные даты и служебные даты. К первой относятся моменты совершения деяния, обнаружения последствий, передачи имущества и другие эпизоды фабулы. Ко второй — прежде всего дата рождения выбранного участника. Третья категория включает даты приговора, постановления, заседания, экспертизы, регистрации и иных процессуальных действий, которые не должны смешиваться с хронологией описываемых фактов.

**Архитектура программного модуля**

Модуль состоит из слабо связанных компонентов. Сервис обработки документов использует общий интерфейс обработчика: TXT-файлы читаются с контролем кодировки, а из DOCX извлекается содержимое абзацев и таблиц через Open XML. Использование специализированного SDK позволяет получать текст без запуска Microsoft Word. Единый интерфейс изолирует парсер от формата источника, а композиционный корень создаёт компоненты и передаёт зависимости, поэтому модуль можно подключать к настольной программе, серверному сервису или пакетному обработчику.

Центральный компонент — класс CrimeDateParser, реализующий интерфейс IDateParser. Он получает нормализованный текст и возвращает ParseResult со списком объектов CrimePeriod, выбранной датой рождения, отклонёнными кандидатами и диагностическими сведениями. CrimePeriod хранит начало и конец интервала; для точной даты значения совпадают. DateCandidate содержит исходное написание, позицию, категорию, оценку контекста и сработавшие правила. Реализация на C# использует строгую типизацию, интерфейсы и развитые средства работы со строками.

**Алгоритм выделения и классификации временных фактов**

Первый этап — нормализация текста. Удаляются невидимые символы, повторяющиеся пробелы, технические номера сносок и разрывы слов; тире и кавычки приводятся к единой форме. Для распознанного текста допускается осторожное исправление ошибок внутри календарного шаблона: замена «О» на «0», восстановление разделителей и объединение разорванного года. Обычные слова при этом не изменяются.

На втором этапе регулярные выражения находят полные числовые даты, формы со словесным месяцем, месяц и год, диапазоны дней или месяцев и приблизительные периоды. Именованные группы позволяют раздельно получить день, месяц, год и границы интервала. Группы, альтернативы и проверки окружения.NET описывают варианты написания без привязки к одному формату. Найденный фрагмент преобразуется в кандидата, но ещё не включается в итоговый список.

Третий этап — анализ контекстного окна: предложения с кандидатом, соседних предложений и ближайших слов. Положительные маркеры фабулы — «совершил», «произошло», «напал», «похитил», «передал», «обнаружен», «эпизод», «деяние», «в период». Отрицательные маркеры отражают процессуальные действия: «приговор», «постановление», «заседание», «экспертиза», «протокол», «решение вступило», «судимость». Каждому правилу назначается вес: близкий положительный признак повышает оценку, отрицательный снижает её. Даты внутри нумерованных эпизодов получают дополнительный приоритет, а реквизиты документа и фразы о вынесении решения указывают на служебную дату. При равенстве признаков кандидат сохраняется как неоднозначный, а не принимается автоматически.

Дата рождения определяется отдельными правилами. Модуль ищет выражения «родился», «дата рождения», «года рождения» и устанавливает, к какому лицу они относятся. Приоритет имеют контексты с ролями «обвиняемый», «подсудимый», «осуждённый» или фамилией выбранного лица. Даты потерпевшего, свидетеля и эксперта не подменяют целевое значение; при нескольких вариантах остальные сохраняются в диагностическом перечне.

Неполные даты нормализуются без создания ложной точности. «Март 1991 года» представляется границами месяца, «в начале марта» — начальным отрезком, а «март–май 1991 года» — периодом от начала марта до конца мая. Восстановление повреждённого года допускается только при однозначном контексте и фиксируется как предположение. После этого удаляются дубли, проверяется календарная допустимость и сохраняется порядок упоминаний в документе.

**Формирование результата и интерпретируемость**

Результат должен быть объяснимым. Вместе с датой сохраняются исходная цитата, номер абзаца, имя файла и правила классификации. Можно установить, что дата принята из-за маркера «совершил» либо исключена из-за сочетания «дата вынесения приговора». Автоматическое решение остаётся проверяемым по первоисточнику.

При пакетной обработке совпадающие факты объединяются, но все ссылки на источники сохраняются. Ошибка чтения отдельного документа не уничтожает уже полученные сведения: сервис возвращает частичный результат и диагностику. Выходная модель может сериализоваться в JSON, записываться в базу данных или передаваться другому компоненту, не связывая парсер с последующими операциями.

**Заключение**

Разработанный программный модуль выполняет получение и нормализацию текста, поиск календарных вы-

ражений, анализ юридического контекста, определение роли участника, обработку точных и интервальных дат, удаление дублей и формирование объяснимого результата. Разделение компонентов по интерфейсам обеспечивает независимость от формата документа и способа дальнейшего использования данных. Подход сокращает объём ручного просмотра, сохраняя смысловую связь даты с фактом и возможность проследить её происхождение до конкретного фрагмента. Дальнейшее развитие может быть связано с выделением участников, мест и правовых оснований, а также с сочетанием интерпретируемых правил и статистических моделей.

Литература:

1. Jurafsky D., Martin J. H. Speech and Language Processing. 2nd ed. Upper Saddle River: Pearson Prentice Hall, 2009. 1024 p.
2. Chalkidis I., Fergadiotis M., Malakasiotis P., Aletras N., Androutsopoulos I. LEGAL-BERT: The Muppets straight out of Law School // Findings of the Association for Computational Linguistics: EMNLP 2020. P. 2898–2904.
3. Manning C. D., Raghavan P., Schütze H. Introduction to Information Retrieval. Cambridge: Cambridge University Press, 2008. 482 p.

## Противодействие DDoS-атакам и веб-угрозам в сегменте официальных интернет-ресурсов уголовно-исполнительной системы

Тюртюбек Георгий Александрович, младший инспектор
УФИЦ ФКУ ИК-4 УФСИН России по Псковской области

*В статье рассматриваются актуальные угрозы информационной безопасности официальных интернет-ресурсов уголовно-исполнительной системы российского государства, связанные с DDoS-атаками и иными веб-угрозами. Автором проводится анализ кибератак на информационные системы Федеральной службы исполнения наказаний, выявляются основные векторы угроз и оценивается эффективность существующих мер противодействия. Особое внимание уделяется проблеме уязвимости официальных сайтов территориальных органов, а также комплексным подходам к защите веб-ресурсов учреждений и органов УИС. На основе изученных материалов и официальных документов ведомства формулируются предложения по совершенствованию системы защиты размещаемых данных на официальных интернет-ресурсах.*

***Ключевые слова:*** *DDoS-атаки, информационная безопасность, уголовно-исполнительная система, официальные интернет-ресурсы, веб-угрозы, ФСИН России, кибербезопасность, защита информации.*

## Counteraction to DDoS attacks and web threats in the segment of official internet resources of the penitentiary system

Turtyubek Georgiy Aleksandrovich, junior inspector
UFIC FKU IK-4 of the Federal Penitentiary Service of Russia for the Pskov Region

*The article considers current information security threats to the official Internet resources of the penal system of the Russian state associated with DDoS attacks and other web threats. An analysis of cyberattacks on the information systems of the Federal Penitentiary Service is conducted, the main threat vectors are identified, and the effectiveness of existing countermeasures is assessed. Special attention is given to the vulnerability of official websites of territorial bodies, as well as to comprehensive approaches to protecting the web resources of institutions and bodies of the penal system. On the basis of studied materials and official documents of the agency, proposals are formulated for improving the system of protection of data placed on official Internet resources.*

***Keywords:*** *DDoS attacks, information security, penitentiary system, official Internet resources, web threats, Federal Penitentiary Service of Russia, cybersecurity, information protection.*

В условиях развития информационного пространства и цифровой трансформации УИС нашего государства официальные интернет-ресурсы ФСИН России и подведомственных учреждений приобретают все большее значение как для действующих сотрудников, так и для лиц, отбывающих наказания, содержащихся под стражей, либо в отношении которых применяется институт пробации. Кроме того, цифровые ресурсы пенитенциарного ведомства являются активными каналами связи и применяются для обмена информацией с сотрудниками, родственниками осужденных, подозреваемых (обвиняемых), обучающихся в учебных заведе-

ниях ФСИН России, выступают инструментами информирования общественности о деятельности УИС, имеют интернет-приемную, различные ссылки и переходы на другие платформы, в том числе для оказания государственных услуг гражданам.

Активное развитие ведомственных информационных систем и их интеграция в единое информационное пространство предусмотрено Концепцией развития уголовно-исполнительной системы Российской Федерации на период до 2030 года. Однако увеличение цифрового присутствия УИС в интернет-пространстве сопровождается ростом угроз информационной безопасности, среди которых особое место занимают DDoS-атаки и иные веб-угрозы, направленные на нарушение доступности и целостности официальных интернет-ресурсов. Для киберпреступников подвергнуть официальный сайт правоохранительных органов или других органов власти DDoS-атакам становится все более распространенным сценарием в условиях нарастающей геополитической напряженности.

Согласно данным сервиса RED Security Anti-DDoS, в третьем квартале 2025 года по России было выявлено и заблокировано почти 50,5 тысяч кибератак, что в 2,3 раза превышает показатели за аналогичный период предыдущего года. При этом эксперты отмечают смещение фокуса внимания хакеров с промышленных предприятий на организации государственного сектора [1].

Уголовно-исполнительная система не является исключением. Как заявил заместитель директора ФСИН России Сергей Щербаков, в 2024 году служба зафиксировала свыше 600 кибератак на свои системы. При этом зарегистрировано более 1 тысячи DDoS-атак и более 10 тысяч различных событий информационной безопасности. По всем инцидентам принимаются меры по их локализации и разбору источников угроз. Кроме того, было зафиксировано 46 попыток внедрения вредоносного программного обеспечения [2].

На основании изученной литературы [3, с. 10–18, 38–51] к угрозам официальным интернет-ресурсам УИС можно применять следующие термины (категории):

DDoS-атаки — распределенные атаки, направленные на перегрузку серверов и нарушение доступности веб-ресурсов. Такие атаки могут существенно нарушить работу критически важных сервисов. Мощность современных DDoS-атак достигает значительных величин: так, самая мощная атака в Уральском федеральном округе, направленная на государственную организацию, достигала 83 Гбит/с.

Атаки на веб-приложения — попытки несанкционированного доступа к информационным системам через уязвимости веб-интерфейсов, включая SQL-инъекции, межсайтовый скриптинг (XSS) и другие методы.

Попытки внедрения вредоносного ПО — направленные на компрометацию информационных систем и получение несанкционированного доступа к конфиденциальной информации.

Комбинированные атаки — использование DDoS-атак для отвлечения внимания специалистов по информационной безопасности в ходе более сложных атак, направленных на взлом инфраструктуры.

Отметим, что DDoS-атаки в современных условиях развития цифрового пространства реже используются злоумышленниками для прямого вымогательства, а чаще — для отвлечения внимания в ходе комплексных атак на информационную инфраструктуру. При этом хакеры активно используют значимые для государства события для проведения атак.

На уязвимость официальных интернет-ресурсов ФСИН России влияет множество факторов. Среди них: 1) высокая общественная значимость информации, размещаемой на официальных сайтах ФСИН России, территориальных органов, ведомственных учебных и научных организациях; 2) необходимость круглосуточной доступности ресурсов для граждан, родственников осужденных, подозреваемых (обвиняемых); 3) интеграция с другими государственными информационными системами; 4) наличие конфиденциальных данных, требующих особого уровня защиты.

Роскомнадзор постоянно фиксирует DDoS-атаки на официальные интернет-ресурсы государственных органов и федеральные информационные системы. Ежедневно хакеры атакуют сервисы РФ, а кибератаки становятся все более мощными. Рост числа кибератак на информационные системы ФСИН России, фиксируемый как на ведомственном уровне, так и в масштабах всего государственного сектора, убедительно свидетельствует о необходимости постоянного совершенствования системы защиты ведомственных веб-ресурсов [4]. Поэтому противодействие DDoS-атакам и веб-угрозам в сегменте официальных интернет-ресурсов УИС требует применения комплексного подхода, включающего организационные, технические и правовые меры.

Для надежной защиты официальных интернет-ресурсов УИС от DDoS-атак предполагаются следующие технические решения.

Специализированные системы защиты от DDoS-атак (Anti-DDoS), которые в автоматическом режиме анализируют трафик в сторону защищаемых ресурсов, выявляют аномальные всплески и обеспечивают фильтрацию вредоносного трафика.

Многоуровневая эшелонированная защита, способная обрабатывать миллиарды событий информационной безопасности в сутки. Как отмечает Минцифры России, для обеспечения информационной безопасности государственных ресурсов выстраивается многоуровневая высокотехнологичная защита.

Web Application Firewall (WAF) — системы, обеспечивающие комплексную защиту веб-приложений от широкого спектра атак.

Системы обнаружения и предотвращения вторжений (IDS/IPS), позволяющие выявлять и блокировать подозрительную активность.

Резервирование инфраструктуры — создание распределенных систем обработки трафика, позволяющих сохранять доступность ресурсов даже при масштабных атаках [5, с. 579–580].

В целях улучшения организационного обеспечения защиты официальных интернет-ресурсов Федеральной службы исполнения наказаний можно предложить проведение круглосуточного мониторинга состояния информационной безопасности с привлечением специально обученных сотрудников УИС. Также целесообразно регулярно обновлять программное обеспечение и оперативно устранять выявленные уязвимости. На постоянной основе следует проводить учения и тренировки IT-специалистов учреждений и органов УИС по противодействию DDoS-атакам и веб-угрозам. Усилить межведомственное взаимодействие в сфере обмена информацией об угрозах и инцидентах между ФСИН России и другими государственными органами, включая Роскомнадзор, Минцифры России и Национальный координационный центр по компьютерным инцидентам.

Подводя итог, подчеркнем, что только при наличии достаточного количества квалифицированных кадров в сфере кибербезопасности уголовно-исполнительная система сможет оперативно реагировать на новые виды цифровых угроз и атак. Создание единой централизованной системы защиты официальных интернет-ресурсов УИС на базе современных Anti-DDoS решений с централизованным управлением и мониторингом позволит противостоять DDoS-атаками и иным веб-угрозам, обеспечит их бесперебойное функционирование и сохранность размещаемой информации. Предлагаемые меры укрепят доверие граждан к цифровой деятельности УИС в целом. Особого внимания заслуживает дальнейшая проработка нормативно-правовых требований к защите официальных интернет-ресурсов УИС, которые отмечаются в трудах современных пенитенциаристов [6, с. 23].

Литература:

1. Новости C-News от 07.11.2025. URL: https://safe.cnews.ru/news/line/2025–11–07_red_security_v_iii_kvartale_2025_g (дата обращения: 21.07.2026).
2. Интервью заместителя директора ФСИН России Сергея Щербакова. РИА Новости от 21.03.2025. URL: https://ria.ru/20250321/scherbakov-2006375882.html (дата обращения: 21.07.2026).
3. Марков А. С. Техническая защита информации. Курс лекций: Учебное пособие для слушателей, обучающихся по курсу «002. Техническая защита информации. Способы и средства защиты информации от несанкционированного доступа». М.: АИСНТ, 2020. 234 с.
4. РИА Новости от 21.03.2025. URL: https://ria.ru/20250321/fsin-2006343573.html (дата обращения: 21.07.2026).
1. Лесько С. А. Модели и методы защиты веб-ресурсов: систематический обзор // Cloud of Science. 2020. Т. 7. № 3. С. 577–610.
2. Ковалев О. Г., Мельникова О. В. Информационные вызовы современного государства и общества и пути их преодоления уголовно-исполнительной системой России // Вестник Владимирского юридического института. 2025. № 3(76). С. 19–24.

## Социальная инженерия и утечки персональных данных как ключевые угрозы информационной безопасности в уголовно-исполнительной системе

Тюртюбек Георгий Александрович, младший инспектор
УФИЦ ФКУ ИК-4 УФСИН России по Псковской области

*В статье рассматриваются актуальные угрозы информационной безопасности уголовно-исполнительной системы Российской Федерации, связанные с применением методов социальной инженерии и утечками персональных данных. Автор анализирует специфику уязвимости УИС перед данными угрозами, основываясь на современной организационной структуре ведомства. Особое внимание уделяется правовым и организационно-техническим аспектам защиты персональных данных осужденных, подозреваемых, обвиняемых и сотрудников УИС. На основе имеющихся статистических данных и материалов прокурорских проверок выявляются системные проблемы в обеспечении информационной безопасности пенитенциарных учреждений. Предлагаются направления совершенствования законодательства и практические меры противодействия социальной инженерии, способы и средства предотвращения утечек конфиденциальной информации.*

***Ключевые слова:*** *социальная инженерия, информационная безопасность, уголовно-исполнительная система, персональные данные, утечки данных, ФСИН России, кибербезопасность, защита информации.*

# Social engineering and personal data leaks as key threats to information security in the penitentiary system

Turtyubek Georgiy Aleksandrovich, junior inspector
UFIC FKU IK-4 of the Federal Penitentiary Service of Russia for the Pskov Region

*The article examines current information security threats to the penal system of the Russian Federation, specifically those arising from the use of social engineering techniques and personal data leaks. The author analyses the specific vulnerabilities of the penal system to these threats, taking into account the contemporary organisational structure of the agency. Particular attention is paid to the legal, organisational and technical aspects of protecting personal data of convicted persons, suspects, accused individuals, and staff of the penal system. On the basis of available statistical data and materials from prosecutors' inspections, systemic problems in ensuring information security of penitentiary institutions are identified. The article proposes directions for improving legislation, practical countermeasures against social engineering, as well as methods and means for preventing leaks of confidential information.*

***Keywords:*** *social engineering, information security, penitentiary system, personal data, data leaks, Federal Penitentiary Service of Russia, cybersecurity, information protection.*

Современный этап развития уголовно-исполнительной системы нашего государства характеризуется активной цифровой трансформацией, внедрением информационных технологий во все сферы деятельности учреждений и органов ФСИН России. Проведение данного вида трансформации и внедрения цифровых технологий деятельность ведомства предусмотрено Концепцией развития уголовно-исполнительной системы Российской Федерации на период до 2030 года.

В современных условиях развития институтов государственной власти цифровизация УИС сопряжена с возникновением новых угроз информационной безопасности, среди которых особое место занимают социальная инженерия и утечки персональных данных. Как справедливо отмечают исследователи [2, с. 9; 3, с. 26], УИС наиболее подвержена цифровым преступлениям в силу особенностей структуры ведомства, и специфики возложенных на нее задач. Взаимодействие со спецконтингентом создает особые условия для совершения преступлений с использованием методов социальной инженерии и несанкционированного доступа к конфиденциальной информации.

В настоящем исследовании под социальной инженерией в УИС будем понимать совокупность психологических методов, используемых злоумышленниками для манипулирования осужденными, подозреваемыми (обвиняемыми), сотрудниками (работниками) пенитенциарного ведомства, получения от них необходимой информации или выполнения тех или иных выгодных для субъекта угрозы действий. В контексте информационной безопасности УИС социальная инженерия — это методы получения необходимого доступа к конфиденциальным сведениям указанных лиц, основанные на их психологических особенностях и специфике деятельности.

Отметим, что специфика УИС создает особые условия для уязвимости перед методами социальной инженерии. Достаточно часто совершение должностных преступлений сотрудниками УИС связано с использованием средств цифровых технологий. Подозреваемые, обвиняемые, осужденные к лишению свободы либо без изоляции от общества совершают преступления повторно также под воздействием данного вида инженерии и при использовании современных Интернет-ресурсов, мессенджеров, социальных сетей.

Все вышеуказанные лица могут становиться объектами манипуляций со стороны злоумышленников, использующих методы социальной инженерии (фишинг, претекстинг, кви-про-кво, иные формы психологического воздействия). Так как пенитенциарные учреждения работают с информацией ограниченного распространения, включающей сведения о лицах, содержащихся под стражей, местах лишения свободы, состоящих на учете уголовно-исполнительной инспекции, сотрудниках режимных объектах, то это повышает привлекательность доступа к таким данным.

Как подчеркивается в исследовательской литературе, для эффективной профилактики и борьбы с цифровыми преступлениями государственные структуры должны быть оснащены эффективными информационно-коммуникационными технологиями [1, с. 145; 4, с. 210], а персонал организации — уметь применять такие технологии.

Рассмотрим еще одну проблему в деятельности учреждений и органов УИС — утечка персональных данных. Киберзащита и информационная безопасность являются приоритетными деятельности территориальных органов ФСИН России.

Кроме того, статистические данные свидетельствуют о масштабности проблемы утечек персональных данных в государственном секторе в целом. Согласно данным компании «Еса Про», за 2025 год 73 % всех утечек данных из российских организаций пришлось на государственный сектор. Всего за отчетный период в сеть утекло более 145 млн строк данных, при этом у государственного сектора было «слито» более 105 млн строк данных с записями о пользователях. Государственные организации стали главной мишенью для хакеров в 2025 году [5].

Прокурорские проверки выявляют нарушения в сфере защиты персональных данных непосредственно в учреждениях УИС. Так, Саянская прокуратура по надзору за

соблюдением законов в исправительных учреждениях проверила исполнение законодательства о персональных данных в ФКУ СИЗО-3 и ФКУ СИЗО-5 ГУФСИН России по Иркутской области. В ходе надзорных мероприятий установлены факты соблюдения не в полной мере требований при обеспечении порядка защиты персональных данных, меры для обеспечения безопасности хранения биометрических персональных данных были недостаточными [6].

Указанные факты свидетельствуют о наличии системных проблем в обеспечении защиты персональных данных в пенитенциарных учреждениях, что требует принятия дополнительных организационно-правовых мер.

Проведенный анализ позволяет выделить следующие ключевые проблемы в сфере противодействия социальной инженерии и предотвращения утечек персональных данных в УИС, а также предложить пути их решения.

В-первую очередь отметим недостаточный уровень цифровой грамотности кадров. Сотрудники и работники УИС должны быть оснащены более мощными технологиями и ресурсами [7, с. 265]. Уровень их осведомленности о методах социальной инженерии остается низким. Считаем, что целесообразно проводить мероприятия воспитательного характера на постоянной основе. Также организовать регулярные занятия с психологом в виде небольших групп. Во-вторых, невысокий уровень кибербезопасности учреждений и органов УИС, а также проблема утечки данных спецконтенгента остается нерешенной. В целях решения данных проблем можем предложить следующее.

1) внедрение систематического обучения сотрудников УИС основам информационной безопасности, тренинги по распознаванию методов социальной инженерии, регулярное проведение учений по противодействию фишинговым атакам и включение соответствующих тем в программы профессиональной подготовки;

2) модернизацию технических средств защиты, внедрение многоуровневой аутентификации, современных антивирусных решений, регулярное обновление программного обеспечения, создание единого защищенного управляемого информационного пространства ФСИН России, использование систем обнаружения вторжений и предотвращения утечек данных;

3) разработку и внедрение единой методики оценки рисков информационной безопасности, создание системы непрерывного мониторинга инцидентов и усиление координации между подразделениями ФСИН России;

4) усиление внутреннего контроля и надзора за соблюдением режима обработки персональных данных, внедрение систем электронного документооборота с автоматической фиксацией всех операций доступа к конфиденциальной информации и ужесточение ответственности за нарушения в сфере защиты персональных данных.

Реализация указанных мер позволит существенно снизить риски утечек персональных данных и повысить устойчивость УИС к угрозам, связанным с социальной инженерией.

Литература:

1. Золотарев М. В. Коррупционная преступность в уголовно-исполнительной системе: современное состояние и проблемы противодействия // Уголовно-исполнительная система: право, экономика, управление. 2019. № 3. С. 141–148.
2. Ковалев О. Г. Роль информационных и цифровых технологий в обеспечении безопасности УИС на современном этапе ее развития // Вестник Томского НИИ ИТ ФСИН России. 2022. № 5. С. 3–10.
3. Ковалев О. Г., Вилкова А. В., Швырев Б. А. Методология формирования знаний об информационной и компьютерной безопасности учреждений УИС // Уголовно-исполнительная система: право, экономика, управление. 2022. № 4. С. 25–27.
4. Миронов Р. Г. Цифровые преступления в уголовно-исполнительной системе: криминологический аспект // Вестник института: преступление, наказание, исправление. 2019. № 4. С. 209–220.
5. Отчет компании «Еса Про» об утечках персональных данных в 2025 году // Коммерсантъ. 22 декабря 2025. URL: https://www.kommersant.ru/doc/8313330 (дата обращения: 21.07.2026).
6. Новости Прокуратуры Иркутской области от 10.02.2025. URL: https://epp.genproc.gov.ru/ru/proc_38/mass-media/news/archive/e7233447/ (дата обращения: 21.07.2026).
7. Мельникова О. В. Искусственный интеллект и цифровая трансформация деятельности уголовно-исполнительной системы // Правоохранительные органы России: проблемы формирования и взаимодействия: Материалы научно-практической конференции с международным участием, Псков, 11–12 ноября 2021 года. Псков: Псковский филиал Академии ФСИН России, 2022. С. 260–265.

# Применение многоадресного вещания для организации взаимодействия компонентов информационных систем

Цветков Александр Александрович, студент
МИРЭА — Российский технологический университет (г. Москва)

*В статье автор исследует возможность применения многоадресного вещания как способа организации обмена сообщениями между составляющими информационной системы. Выполняется создание прототипа информационной системы, состоящей из двух сервисов, использующих возможности сетевого стека операционной системы на базе ядра Linux для обмена сообщениями о событиях посредством многоадресного вещания и выполнении процессов. На основании полученных результатов функционирования прототипа выдвигается предположение о возможности реализации данной концепции, а также предлагается вариант её улучшения.*

***Ключевые слова:*** *сервис-ориентированная архитектура, микросервисная архитектура, сервис, процесс, брокер, многоадресное вещание.*

На сегодняшний день существует множество подходов к созданию информационных систем, отличающихся принципами, фокусом на отдельных составляющих их предметной области, а также способами взаимодействия компонентов. Это обусловлено многообразием решаемых задач и особенностями эксплуатации систем.

В условиях активного внедрения информационных технологий в деятельность компаний к системам предъявляются повышенные требования к скорости работы, безопасности и устойчивости к высоким нагрузкам. Помимо этого, постоянно меняющиеся тренды вынуждают компании предлагать новые и уникальные услуги, тем самым создавая необходимость постоянно дорабатывать свои информационные системы. Именно поэтому в настоящее время акцент при проектировании делается на уменьшение связанности их составляющих, в следствие чего концепция разделения системы на сервисы и выстраивания их взаимодействия получила широкое распространение. Наиболее известной реализацией такой концепции является сервис-ориентированная архитектура (SOA), или же конкретная её разновидность — микросервисная архитектура, отличающаяся от SOA общесистемной стандартизацией способов взаимодействия составляющих систем и возможностью масштабирования за счёт быстрого запуска копий сервисов [1].

В микросервисной архитектуре каждый сервис отвечает за выполнение определенных бизнес-функций, связанных с некоторым элементом предметной области или группой связанных процессов. Безусловно, при проведении большинства процессов в системе, где ответственность разделена, возникает необходимость в межкомпонентном, или говоря иначе, межсервисном взаимодействии. К примеру, обработка заказа в книжном магазине после выбора покупателем позиции к приобретению требует выполнения нескольких процессов — выставление счёта, проведение оплаты и выдача доступа к приобретенному экземпляру после проверки факта оплаты. В дополнение, на почту клиенту может потребоваться отправка чека, что в отношении самого процесса является неприоритетной задачей, но обязательной в соответствии с требованиями законодательства. Именно поэтому существует несколько способов межсервисного взаимодействия в системах. Часто можно наблюдать их комбинацию, так как в рамках процесса одним задачам требуется выполнение действий сторонним сервисом практически мгновенно (синхронно), а другим достаточно выполнения с задержкой (асинхронно).

Наиболее популярным инструментом для организации асинхронного взаимодействия является брокер сообщений. Это специальный программный компонент, который работает по модели «издатель-подписчик». Сервис-издатель публикует некоторое сообщение о событии, а сервис-подписчик реагирует на него в зависимости от определенных в нём бизнес-правил. Бесспорным преимуществом такого компонента является возможность хранения им сообщений, пока они не будут прочитаны потребителем. К тому, же в отличии от синхронной реализации, например, через оповещение о событиях посредством REST API, это позволяет гибко масштабировать систему — добавлять новые узлы-обработчики, что является одним из методов противостояния высоким нагрузкам, или же с минимальными затратами на разработку модифицировать отдельные сервисы, что бывает удобно в системах, которые работают как в «облаке», так и распространяются в коробочном исполнении. Однако использование брокера в системе требует высокой квалификации сопровождающей её персонала, так как только при грамотной настройке будет гарантироваться корректная маршрутизация и низкий процент потерянных сообщений. Это делает их использование в небольших, но нагруженных системах нецелесообразным. К тому, же централизация в такой модели может стать точкой отказа. Конечно, современные брокеры сообщений имеют возможность объединения их в кластеры, однако для небольших систем это становится избыточным.

Исследуя профильное направление компьютерных наук, занимающееся передачей данных, а именно технологий систем связи и компьютерных сетей, можно подметить один из способов передачи данных — многоадресное вещание. Такой способ позволяет эффективно передавать данные множеству получателей одновременно за счёт от-

правки одной копии потока. Для сравнения стоит сказать, что в случае с одноадресной передачей для каждого получателя создаётся своя копия потока, изза чего суммарная нагрузка на источник и сеть линейно растёт с числом клиентов. В качестве еще одного аргумента в пользу многоадресного вещания можно привести возможность управления доставкой потока сетевыми устройствами за счёт протокола IGMP (Internet Group Management Protocol), что избавляет источник данных от необходимости управлять доставкой, а нативная поддержка сетевым стеком операционных систем на базе ядра Linux упрощает использование такого способа передачи данных при разработке прикладных решений [2].

Выполним построение прототипа информационной системы, состоящей из двух сервисов — сервиса задач и сервиса почтовых уведомлений, обменивающихся сообщениями о произошедших в них событиях при помощи многоадресного вещания при проведении некоторого бизнес-процесса создания и делегации задачи. Реализацию сервисов выполним на языке *Python* с использованием библиотеки *multicast.* Данная библиотека позволяет отправлять и получать сообщения по сети посредством многоадресной рассылки, используя возможности сетевого стека операционной системы на ядре Linux [3]. На рис. 1–2 представлены листинги исходного кода сервисов.

```
''' Сервис задач '''

from multicast import send
from multicast import recv
import threading, time, ast

MCAST_ADDR = "224.0.0.1"
MCAST_PORT = 65001

# Модели
class Message:
    def __init__(self, _svc_id, _body):
        self.svc = _svc_id
        self.body = _body

# Функция публикации сообщений
def publisher(msg=None):
    sender = send.McastSAY()
    sender(group=MCAST_ADDR, port=MCAST_PORT, ttl=1, data={"svc": "Tasks", "msg":
msg.body})

# Функция получения сообщений и запуска правил
def listener():
    while True:
        receiver = recv.McastRECV()
        message = receiver(group=MCAST_ADDR, port=MCAST_PORT, ttl=1)
        if message[0]:
            match(ast.literal_eval(message[1])["msg"]):
                case "Email was sent successfully":
                    result = task_delegate()
                    print(result)
                case _:
                   pass
# Бизнес-правила
def task_create():
    while True:
        print("Service 1: New task was created successfully")
        publisher(msg=Message("Tasks", "Task was created successfully"))
        time.sleep(5)

def task_delegate():
    return "Service 1: Task was delegated when the email service sent the event"

l = threading.Thread(target=listener)
p = threading.Thread(target=task_create)

l.start()
p.start()
```

Рис. 1. **Листинг исходного кода сервиса задач с функциями отправки и получения сообщений посредством многоадресного вещания**

```
''' Сервис почтовых уведомлений '''

from multicast import send
from multicast import recv
import threading
import ast

MCAST_ADDR = "224.0.0.1"
MCAST_PORT = 65001

# Модели
class Message:
    def __init__(self, _svc_id, _body):
        self.svc = _svc_id
        self.body = _body

# Функция публикации сообщений
def publisher(msg=None):
    sender = send.McastSAY()
    sender(group=MCAST_ADDR, port=MCAST_PORT, ttl=1, data={"svc": "Mail", "msg":
msg.body})

# Функция получения сообщений и запуска правил
def listener():
    while True:
        receiver = recv.McastRECV()
        message = receiver(group=MCAST_ADDR, port=MCAST_PORT, ttl=1)
        if message[0]:
            match(ast.literal_eval(message[1])["msg"]):
                case "Task was created successfully":
                    result = send_mail()
                    print(result)
                    publisher(msg=Message("Mail", "Email was sent successfully"))
                case _:
                   print("Invalid event ID")


def send_mail():
    return "Service 2: Email was sent successfully"

listener = threading.Thread(target=listener)

listener.start()
```

Рис. 2. **Листинг исходного кода сервиса почтовых уведомлений с функциями отправки и получения сообщений посредством многоадресного вещания**

Программы используют многопоточность для разделения задач отправки и обработки получаемых сообщений. После запуска программы сервиса задач выполняется создание задачи посредством функции *task_create()*, в которой также вызывается функция *publisher()*, отправляющей сообщение «*Task was created successfully*» в сеть по адресу многоадресного вещания *224.0.0.1*. Запущенная программа сервиса почтовых уведомлений прослушивает данный адрес и при поступлении сообщений начинает их обработку — выполняет чтение сообщения и запуск соответствующего бизнес-сценария. В данном примере таким сценарием является отправка оповещения на почту посредством функции *send_mail()* и публикация сообщения «*Email was sent successfully*» в сеть по адресу многоадресного вещания *224.0.0.1*. В сервисе задач также задано правило обработки события отправки письма, и поэтому при получении данного сообщения из сети сервис задач выполняет бизнес-сценарий делегации задачи, реализованной в функции *task_delegate()*. На рис. 3. представлено логирование обмена сообщениями и отработки бизнес-сценариев сервисами.

```
(.venv) sasha@sasha-asus:~/miniIS$ python3 service1.py
Service 1: New task was created successfully

{'svc': 'Tasks', 'msg': 'Task was created successfully'}


{'svc': 'Mail', 'msg': 'Email was sent successfully'}

Service 1: Task was delegated when the email service sent the event
```

```
(.venv) sasha@sasha-asus:~/miniIS$ python3 service2.py

{'svc': 'Tasks', 'msg': 'Task was created successfully'}

Service 2: Email was sent successfully

{'svc': 'Tasks', 'msg': 'Task was created successfully'}

Service 2: Email was sent successfully
```

Рис. 3. **Логирование обмена сообщениями и отработки бизнес-сценариев сервисами**

Нельзя не упомянуть о главном недостатке подобного исполнения — потеря сообщений при недоступности получателя. Дело в том, что многоадресное вещание изначально разрабатывалось для сервисов реального времени — например, IPTV (Internet Protocol Television) и сохранение состояния не было предусмотрено осознано. Одним из решений данной проблемы, вызванной ограничением многоадресного вещания, может стать добавление кэширующих компонентов в составе сервиса-подписчика (рис. 4). При недоступности всех экземпляров сервиса-подписчика сообщения принимаются кэширующим компонентом сервиса и при восстановлении работоспособности первый экземпляр сервиса выполняет обработку имеющихся в кэше данных.

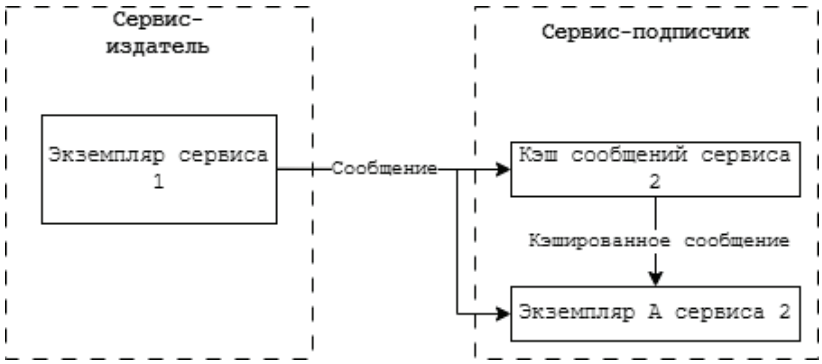


Рис. 4. **Схема обмена сообщениями при использовании кэширующего компонента**

Подводя итог, можно предположить, что концепция использования многоадресного вещания имеет место быть при создании небольших, но нагруженных или аппаратно-разделенных информационных систем за счёт своей эффективности и простоты реализации. В дополнение можно сказать, что создаваемые информационные системы с применением многоадресного вещания как способа взаимодействия между их составляющими могут быть включены в состав автоматизированных систем предприятия, реализующих технологические процессы, так как некоторые программируемые логические контроллеры умеют получать данные, передаваемые многоадресным вещанием.

Литература:

1. SOA vs. Микросервисы: в чём отличия. — Текст: электронный // Джеймикс: [сайт]. — URL: https://www.jmix.ru/tech-hub/soa-vs-mikroservisy/ (дата обращения: 19.07.2026).
2. IP Multicast Routing Technology Overview. — Текст: электронный // Cisco Systems: [сайт]. — URL: https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst9300/software/release/ 17–1/configuration_guide/ip_mcast_rtng/b_171_ip_mcast_rtng_9300_cg/ip_multicast_routing___technology_overview.pdf (дата обращения: 19.07.2026).
3. Multicast Python Module. — Текст: электронный // PyPI: [сайт]. — URL: https://pypi.org/project/multicast/ (дата обращения: 19.07.2026).

1

# Молодой ученый

Международный научный журнал

№ 30 (633) / 2026

Выпускающий редактор Г. А. Письменная

Ответственные редакторы Е. И. Осянина, О. А. Шульга, З. А. Огурцова

Художник Е. А. Шишков

Подготовка оригинал-макета П. Я. Бурьянов, М. В. Голубцов, О. В. Майер



Журнал размещается и индексируется на портале eLIBRARY.RU, на момент выхода номера в свет журнал не входит в РИНЦ.

Свидетельство о регистрации СМИ ПИ № ФС77-38059 от 11 ноября 2009 г., выдано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзор).

ISSN-L 2072-0297

ISSN 2077-8295 (Online)

Учредитель и издатель: ООО «Издательство Молодой ученый». 420029, Республика Татарстан, г. Казань, ул. Академика Кирпичникова, д. 25, пом. 1, 3, 4, 5, 6.

Номер подписан в печать 05.08.2026. Дата выхода в свет: 12.08.2026.

Формат 60×90/8. Тираж 500 экз. Цена свободная.

Почтовый адрес редакции: 420140, Республика Татарстан, г. Казань, ул. Юлиуса Фучика, д. 94А, а/я 121.

Фактический адрес редакции: 420029, Республика Татарстан, г. Казань, ул. Академика Кирпичникова, д. 25, пом. 1, 3, 4, 5, 6.

E-mail: info@moluch.ru; https://moluch.ru/

Отпечатано в типографии издательства «Молодой ученый», 420029, Республика Татарстан, г. Казань, ул. Академика Кирпичникова, д. 25, пом. 1, 3, 4, 5, 6.