Разработка и анализ матричных структур для оптимизации хранения и обработки типовых конфигурационных данных в учетно-управленческих системах 1с

Попова Виктория Алексеевна
ассистент кафедры алгебраических и информационных систем, Иркутский государственный университет; ассистент кафедры высшей математики, Санкт-Петербургский государственный университет путей сообщения Императора Александра I, victoryuropova1@gmail.com

Достоверность экономической информации предприятия во многом определяется надежностью учетно-управленческих систем, значительная часть которых в России построена на платформе «1С:Предприятие». В работе рассматривается задача эффективного хранения и обработки типовых конфигурационных данных 1С – сведений о типах, порождаемых объектами метаданных, и о правилах их совместимости, используемых инструментами статического анализа. Показано, что плотные матрицы совместимости размерности |T|×|T| неприемлемы для крупных конфигураций, поскольку объем памяти растет квадратично.
Предложена факторизованная матричная структура (ФМС), включающая разбиение типов на классы по виду и роли, компактную матрицу классов над алфавитом {0, 1, δ} и разреженное множество исключений; проверки составных типов сведены к операциям булевой матричной алгебры. Вычислительный эксперимент на моделях конфигураций от 500 до 10 000 объектов показал, что объем ФМС не превышает 0,3 МБ против 1,1 ГБ для битовой и 182 МБ для разреженной (CSR) матрицы, а проверка крупных составных типов ускоряется на два порядка. Предложена модель оценки экономического эффекта от применения структуры.
Ключевые слова: 1С:Предприятие, типовые конфигурационные данные, матрица типовой совместимости, факторизованная матричная структура, разреженные матрицы, булева матричная алгебра, составные типы, статический анализ, экономические информационные системы, экономическая эффективность.

Исследование выполнено за счет гранта Российского научного фонда (проект № 25-11-00348).

Введение

Учетно-управленческие системы на платформе «1С:Предприятие» используются для ведения бухгалтерского, налогового и управленческого учета, расчета заработной платы, управления закупками, продажами и производством. Ошибки в программном коде таких систем непосредственно отражаются на достоверности экономической информации: искажают учетные данные и отчетность, вызывают простои бизнес-процессов и прямые финансовые потери. Стоимость исправления ошибки, обнаруженной при эксплуатации, значительно превышает стоимость ее устранения на этапе разработки [1–4, 18], поэтому инструментальные средства статического анализа имеют не только техническое, но и экономическое значение.

В предыдущих работах автора [5–11] предложен подход к статической проверке типов для встроенного языка «1С:Предприятия», основанный на «Дереве типов конфигурации» (ДТК) и матрицах типовой совместимости, задающих допустимость операций для каждой пары типов. При переходе к крупным конфигурациям узким местом становится хранение самих типовых данных. Каждый объект метаданных порождает несколько типов: справочник «Контрагенты» – типы СправочникСсылка.Контрагенты, СправочникОбъект.Контрагенты, СправочникМенеджер.Контрагенты, СправочникВыборка.Контрагенты и СправочникСписок.Контрагенты. В результате для конфигурации из 5000 объектов плотное хранение четырех матриц совместимости требует более 2 ГБ оперативной памяти.

Цель работы – разработка и анализ матричных структур, обеспечивающих компактное хранение и быструю обработку типовых конфигурационных данных 1С без потери точности проверки, а также оценка экономического эффекта от их применения.

Типовые конфигурационные данные и постановка задачи

Под типовыми конфигурационными данными понимается совокупность сведений о типах конфигурации, используемых анализатором кода [15]: ДТК [5], описания типов реквизитов и параметров (в том числе составных), матрицы совместимости операций и таблицы результирующих типов. Обозначим через T множество типов конфигурации, через T_P – множество встроенных типов платформы, через n_v – число объектов вида $v \in V$ (справочники, документы, регистры и т. д.), через r_v – число типов, порождаемых одним объектом вида v (роли «Ссылка», «Объект», «Менеджер» и т. п.). Тогда

$$|T| = |T_P| + \sum_{v \in V} r_v n_v. \qquad (1)$$

Для каждой операции op из множества O задается матрица $M_{op}: T \times T \to \{0,1\}$. При плотном хранении (1 байт на элемент) и при битовой упаковке объем составляет

$$V_{\text{плот}} = |O| \cdot |T|^2, \quad V_{\text{бит}} = \frac{|O| \cdot |T|^2}{8}. \qquad (2)$$

Поскольку r_v в среднем близко к пяти, при 10 000 объектов $|T| \approx 48\,000$, и даже битовая упаковка четырех матриц требует около 1,1 ГБ. Кроме того, структура хранения должна обеспечивать постоянное время проверки пары типов, проверку составных типов без перебора всех пар и обновление при изменении метаданных без полного перестроения.

Структурные свойства матриц совместимости

Если упорядочить типы в порядке обхода ДТК в глубину, то типы одного вида и роли (например, СправочникСсылка.*) образуют непрерывный интервал номеров, а матрица распадается на блоки трех видов (рис. 1, а): нулевые (операция недопустима, например сложение ссылок), единичные (допустима для любой пары, например сравнение на равенство ссылок разных видов для реквизита «Регистратор») и диагональные, или δ-блоки (допустима только для совпадающих типов, например присваивание ссылки реквизиту, типизированному ссылкой на конкретный справочник). Помимо блоков встречаются немногочисленные нерегулярные правила – исключения. Заполненность матриц сильно различается: матрица сложения содержит единицы лишь в нескольких элементах, а матрица равенства – плотный блок ссылочных типов. Поэтому универсальные разреженные форматы [13, 14] эффективны только для части матриц.

Факторизованная матричная структура

Пусть K – множество классов типов (пар «вид – роль» и отдельных встроенных типов), $c: T \to K$ – отображение типа в класс, $k = |K|$. ФМС для операции op задается тройкой (c, C_{op}, E_{op}), где $C_{op}: K \times K \to \{0,1,\delta\}$ – матрица классов, $E_{op} \subset T \times T$ – множество исключений. Элемент исходной матрицы восстанавливается по правилу

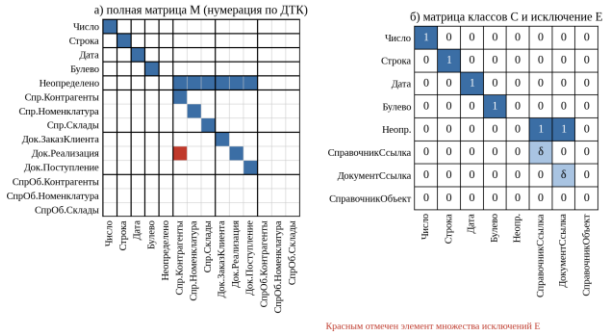


Рис. 1. Блочная структура матрицы совместимости операции присваивания: а) полная матрица; б) матрица классов

$$M_{op}[i,j] = [C_{op}[c(i),c(j)] = 1] \vee ([C_{op}[c(i),c(j)] = \delta] \wedge [i = j]) \vee [(i,j) \in E_{op}], \quad (3)$$

где $[\cdot]$ – индикатор истинности условия (рис. 1, б). Отображение c общее для всех операций и хранится массивом 16-разрядных номеров классов длиной $|T|$; элемент матрицы классов кодируется двумя битами, исключения хранятся в хеш-множестве. Объем ФМС в байтах составляет

$$V_{\text{ФМС}} = 2|T| + \frac{|O| \cdot k^2}{4} + 16 \sum_{op \in O} |E_{op}|. \quad (4)$$

Первое слагаемое линейно по числу типов, второе не зависит от размера конфигурации (число классов – несколько сотен), третье пропорционально числу нерегулярных правил: квадратичная зависимость от $|T|$ заменяется линейной. Алгоритм проверки пары типов (рис. 2) выполняется за постоянное время: два обращения к массиву классов, одно – к матрице классов, целиком помещающейся в кэш процессора, и при необходимости одно – к хеш-множеству. Структура встраивается в матричный движок анализатора [8–10] через ту же функцию проверки совместимости, что и плотная матрица.

```
Функция Совместимы(ФМС, Операция, i, j)
    МатрицаКлассов = ФМС.МатрицыКлассов[Операция];
    v = МатрицаКлассов[ФМС.Класс[i]][ФМС.Класс[j]];
    Если v = 1 Тогда
        Возврат Истина;          // единичный блок
    ИначеЕсли v = 2 И i = j Тогда
        Возврат Истина;          // δ-блок, типы совпадают
    КонецЕсли;
    Ключ = i * ФМС.ЧислоТипов + j;
    Возврат ФМС.Исключения[Операция].Получить(Ключ) <> Неопределено;
КонецФункции
```

Рис. 2. Проверка совместимости пары типов в ФМС

Обработка составных типов

Реквизиты в 1С могут иметь составной тип; реквизиты вида «Регистратор» и определяемые типы включают сотни ссылочных типов. Представим составной тип булевым вектором $x \in \{0,1\}^{|T|}$ и обозначим через $\otimes$ булево произведение матрицы на вектор: $(M \otimes y)_i = \bigvee_j (M_{ij} \wedge y_j)$. Три режима проверки – допустимость хотя бы одной комбинации типов ($\exists\exists$), строгая проверка ($\forall\forall$) и проверка присваивания, когда каждый тип значения должен быть допустим хотя бы для одного типа приемника ($\forall\exists$), – записываются так:

$$\exists\exists: x^T \otimes M \otimes y = 1, \forall\forall: x^T \otimes \neg M \otimes y = 0, \forall\exists: x \wedge \neg(M \otimes y) = 0. \quad (5)$$

На плотной матрице это требует $O(|x| \cdot |y|)$ операций. В ФМС вектор y проецируется на классы: $\hat{y}_c = \bigvee_{i:\, c(i)=c} y_i$, и с учетом (3)

$$(M \otimes y)_i = (C_{op} \otimes \hat{y})_{c(i)} \vee (d_{c(i)} \wedge y_i) \vee (E_{op} \otimes y)_i, \quad (6)$$

где в произведении $C_{op} \otimes \hat{y}$ элементы δ считаются нулевыми, $d_c = [C_{op}[c,c] = \delta]$. Трудоемкость составляет $O(|y| + k \cdot |\hat{y}| + |x| \cdot log|y|)$, где $|\hat{y}|$ – число различных классов (как правило, единицы), т. е. определяется числом классов, а не числом типов; это соответствует выражению алгоритмов на графах через линейную алгебру над булевым полукольцом [16].

Рассмотрим пример (рис. 3). Здесь x = {ДокументСсылка.ЗаказКлиента, ДокументСсылка.ВозвратТоваров}, y = {ДокументСсылка.ЗаказКлиента, ДокументСсылка.РеализацияТоваров}, проекция $\hat{y}$ = {ДокументСсылка}, а C_{ASSIGN}[ДокументСсылка, ДокументСсылка] = δ. Тип ЗаказКлиента входит в y и допустим; тип ВозвратТоваров в y не входит и исключений для него нет, поэтому формируется диагностическое сообщение (рис. 4).

Инкрементальное обновление

В плотном представлении добавление объекта вида v требует перевыделения памяти и копирования $O(|T|^2)$ элементов каждой матрицы. В ФМС оно сводится к добавлению r_v элементов в массив классов, т. е. выполняется за $O(r_v)$: новый тип наследует правила своего класса. Непрерывность интервалов сохраняется двухуровневой адресацией (класс, локальный номер) без перенумерации типов. Изменение правила для класса сводится к изменению одного элемента C_{op}. Для конфигурации из 10 000 объектов ФМС занимает менее 300 КБ и может храниться в кэше проекта 1С:EDT [12].

```
// Реквизит «Основание» документа ПоступлениеОплаты имеет составной тип:
//   ДокументСсылка.ЗаказКлиента, ДокументСсылка.РеализацияТоваров
Процедура ЗаполнитьОснование(ДокументОбъект, Источник)
    // Выведенный тип параметра Источник: ДокументСсылка.ЗаказКлиента,
    //                                    ДокументСсылка.ВозвратТоваров
    ДокументОбъект.Основание = Источник;
КонецПроцедуры
```

Рис. 3. Присваивание значения составного типа

```
ПРЕДУПРЕЖДЕНИЕ ТИПИЗАЦИИ: присваивание значения составного типа
Местоположение: Модуль документа "ПоступлениеОплаты", строка 6
Приемник: реквизит Основание
  Допустимые типы: ДокументСсылка.ЗаказКлиента,
                   ДокументСсылка.РеализацияТоваров
Типы значения, недопустимые для приемника:
  • ДокументСсылка.ВозвратТоваров
Правило: C_ASSIGN[ДокументСсылка, ДокументСсылка] = δ
        (совместимы только совпадающие типы)
Проверено классов: 1, типов: 2
```

Рис. 4. Сгенерированный диагностический отчет

Вычислительный эксперимент

Модель конфигурации включает 400 встроенных типов платформы и десять видов объектов: справочники, документы, планы видов характеристик и счетов порождают по пять типов, перечисления – три, регистры сведений – семь, регистры накопления – шесть, обработки, отчеты и константы – по два. Доли видов: справочники – 30 %, документы – 15 %, перечисления – 15 %, регистры сведений – 20 %, регистры накопления – 6 %, обработки – 7 %, отчеты – 4,5 %, прочие – 2,5 %; всего $k = 442$ класса. Рассматривались матрицы сложения, сравнения, равенства и присваивания (для последней генерировались исключения в количестве 0,2 % от числа объектов). Сравнивались плотная матрица, битовая упаковка, формат CSR [14] и ФМС. Корректность ФМС проверена сравнением с плотной матрицей на 10^6 случайных пар для всех операций при размерах до 2000 объектов; расхождений не выявлено. Результаты по памяти приведены в табл. 1 и на рис. 5.

Таблица 1
Объем памяти для хранения четырех матриц совместимости, МБ

Число объектов	Число типов |T|	Плотная	Битовая	CSR	ФМС
500	2 791	29,7	3,71	0,536	0,195
1 000	5 185	102,6	12,8	1,97	0,200
2 000	9 970	379,2	47,4	7,53	0,209
5 000	24 325	2 257,2	282,1	45,9	0,236
10 000	48 250	8 880,9	1 110,1	181,9	0,282

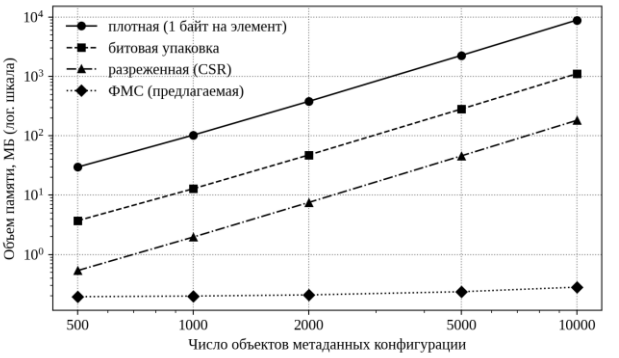


Рис. 5. Зависимость объема памяти от размера конфигурации

Объем ФМС практически постоянен (0,195–0,282 МБ); основная его часть приходится на матрицы классов. Для 10 000 объектов выигрыш составляет более 31 000 раз по сравнению с плотной матрицей, около 3900 раз – с битовой и около 640 раз – с CSR; последний проигрывает из-за плотного блока матрицы равенства (около 38 млн ненулевых элементов). Временные характеристики (Python 3.12 / NumPy 2.4, одноядерная виртуальная машина, минимум по серии запусков) приведены в табл. 2 и 3.

Таблица 2
Временные характеристики операций с матрицей равенства, мс

Операция	Формат	500 объектов	1 000 объектов	2 000 объектов
Построение из правил	Плотная	36,2	137,3	427,1
	ФМС	< 0,01	< 0,01	< 0,01
Добавление объекта	Плотная	2,1	8,1	24,2
	ФМС	< 0,01	< 0,01	< 0,01
Проверка 10^6 пар типов	Плотная	7,0	7,7	17,5
	Битовая	10,8	9,9	11,1
	CSR	16,8	18,1	19,9

Операция	Формат	500 объектов	1 000 объектов	2 000 объектов
	ФМС	7,9	8,0	7,7

Таблица 3
Время проверки присваивания составного типа (режим ∀∃, 2000 объектов), мс

Число типов в составном типе	Плотная	ФМС	Отношение
10	0,007	0,054	0,13
100	0,050	0,079	0,63
1 000	3,77	0,070	54
1 230 (все ссылочные типы)	5,75	0,050	115

Построение и обновление ФМС выполняются на несколько порядков быстрее, чем плотной матрицы. Время пакетной проверки пар для ФМС не зависит от размера конфигурации, тогда как для плотной матрицы оно растет из-за промахов кэша. Для матрицы присваивания с исключениями ФМС пока уступает плотной (38,1 мс против 21,0 мс на 10^6 проверок); разрыв может быть сокращен хранением признака наличия исключений в блоке. Для небольших составных типов плотная матрица выигрывает за счет меньших накладных расходов, а при сотнях типов ФМС быстрее в 50–115 раз, поэтому целесообразен гибридный режим: формула (3) для малых составных типов, формула (6) – для крупных.

Оценка экономического эффекта

Применение ФМС дает эффект по двум направлениям: снижение затрат на вычислительные ресурсы и снижение затрат на исправление ошибок, которые без анализа типов были бы обнаружены только при эксплуатации. Годовой эффект оценивается выражением

$$Э = Э_{инф} + Э_{ош} = N_р \cdot \Delta V \cdot c_V + N_{ош} \cdot (c_э - c_р), \qquad (7)$$

где $N_р$ – число рабочих мест и серверов непрерывной интеграции, выполняющих анализ; ΔV – сокращение потребности в памяти на одно место (по табл. 1 – более 2 ГБ при 5000 объектов для плотного хранения); c_V – удельная годовая стоимость памяти; $N_{ош}$ – число ошибок типизации в год, выявляемых при разработке; $c_э$ и $c_р$ – средняя стоимость исправления ошибки при эксплуатации и при разработке [1, 18]. Основной вклад, как правило, вносит второе слагаемое: при плотном хранении анализ крупных конфигураций на рабочем месте разработчика практически невыполним, и ФМС снимает это ограничение. Параметры модели определяются по данным предприятия: журналам инцидентов, трудозатратам на сопровождение и стоимости инфраструктуры.

Сравнение с существующими подходами и ограничения

Битовая упаковка сохраняет квадратичную асимптотику; разреженные форматы [13, 14] не учитывают блочную структуру; двоичные решающие диаграммы [17] компактны, но их размер зависит от порядка переменных, а операции сложнее в реализации. ФМС использует априорное знание о системе типов платформы, и каждое решение анализатора прослеживается до элемента матрицы классов или исключения. Эффективность ФМС снижается при большом числе нерегулярных правил; кроме того, результаты получены на синтетических моделях, и их подтверждение на реальных конфигурациях является предметом дальнейшей работы.

Заключение

Предложена факторизованная матричная структура для хранения и обработки типовых конфигурационных данных учетно-управленческих систем 1С, основанная на разбиении типов на классы, матрице классов над алфавитом {0, 1, δ} и множестве исключений. Получены формулы проверки простых и составных типов в терминах булевой матричной алгебры. Показано, что объем памяти ФМС растет линейно с малым коэффициентом (0,28 МБ против 1,1 ГБ для битовой матрицы при 10 000 объектов), обновление выполняется за время, пропорциональное числу добавляемых типов, а проверка крупных составных типов ускоряется на два порядка. Модель оценки экономического эффекта показывает, что основная выгода связана с возможностью применять статический анализ к крупным системам и выявлять ошибки до ввода в эксплуатацию. Перспективы работы – апробация на реальных конфигурациях с оценкой эффекта по данным предприятий и распространение подхода на таблицы результирующих типов.

Литература

1. Коданев В. Л., Федин Ф. О., Грачёва Е. В. Технология интеграции и тестирования автоматизированных систем // Современные наукоемкие технологии. 2024. № 10. С. 100-104.
2. Раджабова Ф. М. Алгоритмы обнаружения и исправления ошибок в программном коде // Актуальные вопросы современной экономики. – 2025. – № 4. – С. 791-793.
3. Цифровые технологии в решении проблем современности: монография / Р. С. Зарипова, Ю. С. Валеева, Ю. Н. Смирнов [и др.]. Казань: Казанский государственный энергетический университет, 2023. 298 с.
4. Абрикосов К., Попова В. А. Статическая типизация в веб-приложениях на Node.js: процесс внедрения и выявление ошибок несоответствия типов // Информационные технологии и математическое моделирование в управлении сложными системами. 2023. № 2(18). С. 23-32.
5. Балюк А. С., Попова В. А. Разработка программного комплекса для конвертации конфигурации платформы "1С:Предприятие" в UML-модель // Вестник Российского нового университета. Серия: Сложные системы: модели, анализ и управление. 2021. № 4. С. 137-145.
6. Balyuk A. S., Popova V. A. Static type-checking for programs developed on the platform 1C:Enterprise // CEUR Workshop Proceedings : 4, Irkutsk, 14 September 2021. Irkutsk, 2021. P. 101-111.
7. Попова В. А. Улучшение качества программного обеспечения на платформе «1С:Предприятие» при помощи статической проверки типов // Комбинаторные и вероятностные свойства дискретных структур : Сборник научных трудов. Иркутск : Иркутский государственный университет, 2025. С. 87-92.
8. Попова В. А. Разработка инструментов для автоматического нахождения ошибок статической проверки типов в конфигурациях "1С:Предприятия" // Новые информационные технологии в образовании : Сборник научных трудов XXIV Международной научно-практической конференции, Москва, 30–31 января 2024 года. Москва: ООО "1С-Паблишинг", 2024. С. 184-186.
9. Попова В. А. Разработка правил для выявления ошибок несоответствия типов выражений в языке программирования "1С: Предприятие" // Динамические системы и компьютерные науки: теория и приложения (DYSC 2023) : материалы 5-й Международной конференции, Иркутск, 18–23 сентября 2023 года. Иркутск: Иркутский государственный университет, 2023. С. 169-172.
10. Попова, В. А. Проектирование механизма статического анализа для выявления ошибок несоответствия типов в программах на динамических языках программирования // Динамические системы и компьютерные науки: теория и приложения (DYSC 2022) : Материалы 4-й Международной конференции, Иркутск, 19–22 сентября 2022 года. Иркутск: Иркутский государственный университет, 2022. С. 142-145.
11. Попова, В. А. Применение методики статического анализа для выявления ошибок в программах на динамических языках программирования // Информационные технологии. Проблемы и решения. 2022. № 2(19). С. 26-31.
12. 1C:Enterprise Development Tools. URL: https://its.1c.ru/db/edtdoc (дата обращения: 09.11.2025).
13. Писсанецки С. Технология разреженных матриц / пер. с англ. М.: Мир, 1988. 410 с.
14. Saad Y. Iterative Methods for Sparse Linear Systems. 2nd ed. Philadelphia: SIAM, 2003. 528 p.
15. Ахо А. В., Лам М. С., Сети Р., Ульман Дж. Д. Компиляторы: принципы, технологии и инструментарий. 2-е изд. М.: Вильямс, 2008. 1184 с.
16. Kepner J., Gilbert J. (eds.) Graph Algorithms in the Language of Linear Algebra. Philadelphia: SIAM, 2011.
17. Bryant R. E. Graph-Based Algorithms for Boolean Function Manipulation // IEEE Transactions on Computers. 1986. Vol. C-35, No. 8. P. 677–691.
18. Boehm B. W. Software Engineering Economics. Englewood Cliffs: Prentice-Hall, 1981. 767 p.

Development and analysis of matrix structures for optimizing storage and processing of configuration type data in 1c accounting and management systems

Popova V.A.
Irkutsk State University, Emperor Alexander I St. Petersburg State Transport University

JEL classification: C01, C02, C1, C4, C5, C6, C8

The reliability of an enterprise's economic information largely depends on its accounting and management systems, a significant share of which in Russia are built on the 1C:Enterprise platform. The paper addresses efficient storage and processing of 1C configuration type data, i.e. information about the types generated by metadata objects and about their compatibility rules used by static analysis tools. It is shown that dense |T|×|T| compatibility matrices are infeasible for large configurations, since memory consumption grows quadratically.

A factorized matrix structure (FMS) is proposed, comprising a partition of types into classes by kind and role, a compact class matrix over the alphabet {0, 1, δ}, and a sparse set of exceptions; checks for composite types are reduced to Boolean matrix algebra operations. A computational experiment on configuration models with 500 to 10,000 objects showed that the FMS takes at most 0.3 MB, compared with 1.1 GB for a bit-packed and 182 MB for a sparse CSR matrix, while checks of large composite types are accelerated by two orders of magnitude. A model for estimating the economic effect of the structure is proposed.

Keywords: 1C:Enterprise, configuration type data, type compatibility matrix, factorized matrix structure, sparse matrices, Boolean matrix algebra, composite types, static analysis, economic information systems, economic efficiency.

References

1. Kodanev V. L., Fedin F. O., Gracheva E. V. Technology of integration and testing of automated systems // Modern High-Tech Technologies. 2024, no. 10, pp. 100-104.
2. Radzhabova F. M. Algorithms for detection and correction of errors in program code // Topical Issues of Modern Economics. 2025, no. 4, pp. 791-793.
3. Zaripova R. S., Valeeva Yu. S., Smirnov Yu. N. [et al.]. Digital Technologies in Solving Contemporary Problems: monograph. Kazan: Kazan State Power Engineering University, 2023. 298 p.
4. Abrikosov K., Popova V. A. Static typing in Node.js web applications: implementation process and detection of type mismatch errors // Information Technologies and Mathematical Modeling in the Control of Complex Systems. 2023, no. 2(18), pp. 23-32.
5. Balyuk A. S., Popova V. A. Development of a software package for converting the "1C:Enterprise" platform configuration into a UML model. // Bulletin of the Russian New University. Series: Complex Systems: Models, Analysis and Management. 2021, no. 4, pp. 137-145.