ИТ-интегратор «Белый Код»

30013, г. Тула, ул. Болдина, д. 47 А, оф. 14
тел. 8-800-777-54-01, 8(4872)75-11-02,
e-mail: lobanovann@bpca.ru, сайт: bpca.ru

Отчет о проверке конфигурации для компании Ритейл.

Вводные данные

Конфигурация: Управление торговлей, редакция 11 (11.5.11.66) с доработками.
Источник на сервере компании: Srvr="sdsrv5";Ref="ut11_dev2";

Хранилище не используется, данные об авторах изменений отсутствуют.
Объекты на поддержке вендора, с запретом изменения исключены из проверки.

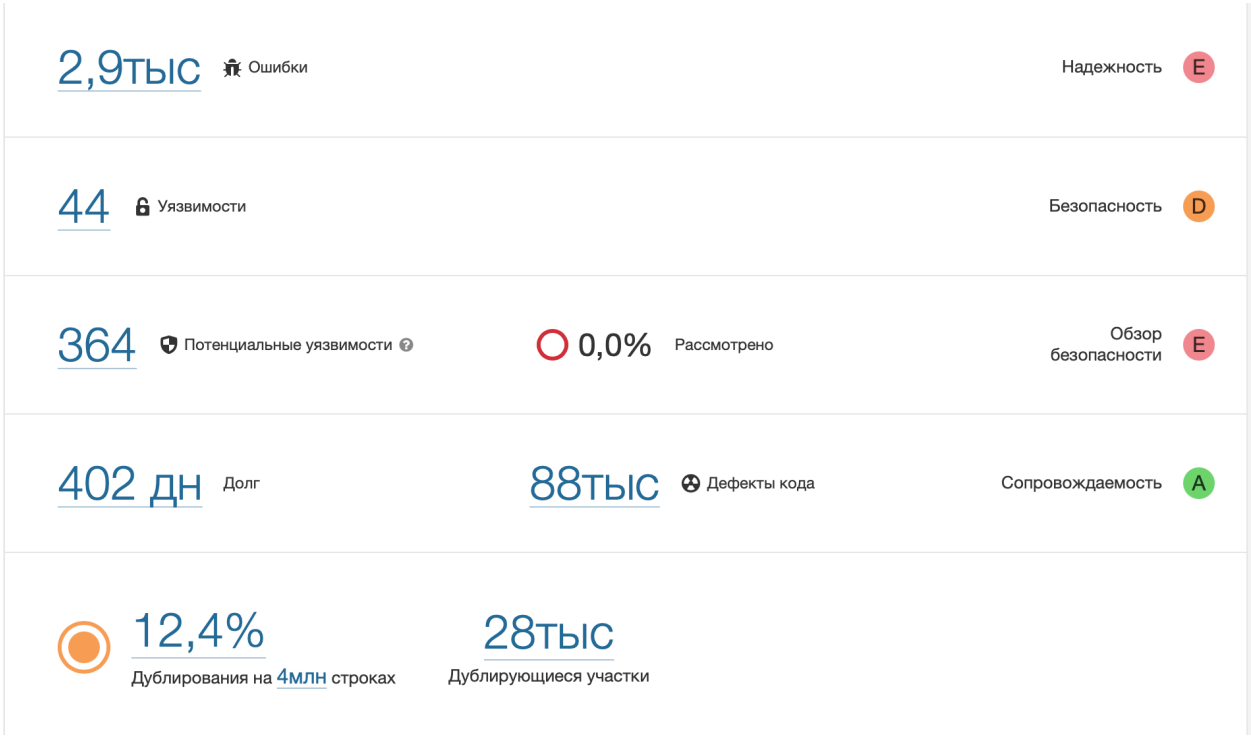
Для проверки использовался SonarQube 1C (BSL) Community Plugin
Результаты размещены на сервере SonarQube____ (будет ссылка на ваш отчет)

Логин: _____
Пароль: _____

Описание используемых терминов

Термин	Описание
Ошибка	Замечание, которое указывает, что-то не так в коде. Даже если код сейчас работает, есть вероятность что он сломается в самый неподходящий момент.
Дефекты кода	Замечание, связанная со сложностью поддержки кода. Большое количество дефектов как минимум усложнит работу разработчикам, поддерживающим код. В худшем случае они будут вынуждены вместо внесения изменений полностью переписать код заново.
Долг	См. технический долг.
Замечание	Когда фрагмент кода не соответствует правилу, регистрируется замечение. Существует 3 типа замечаний: ошибки, дефекты кода и уязвимости.
Правило	Стандарты и правила кодирования.
Технический долг	Предполагаемое время, необходимое для устранения всех замечаний. Измеряется в минутах. При пересчете в дни используется 8 часовой рабочий день.
Уязвимость	Замечание, связанное с безопасностью, которая потенциально может быть искоторая представляет собой бэкдор для злоумышленников.

Общий обзор показателей



Обзор ошибок

К ошибкам относятся проблемы способные привести к выбросу исключения во время работы конфигурации или радикальной потери производительности.

2 956 замечаний, классифицировано как ошибки

Модуль	Важность	Итого	BLOCKER	CRITICAL	MAJOR	MINOR
cf/Reports/ИА_ВедомостьПоФондам/Forms/ФормаОтчета/Ext/Form/Module.bsl		**402**	3	-	399	-
cf/CommonModules/КСУ_ОргсхемаДлительныеОперации/Ext/Module.bsl		**109**	-	47	50	12
cf/Catalogs/ОргСхема/Forms/ФормаВыбораСтандартнойСхемыСоздание/Ext/Form/Module.bsl		**107**	-	47	48	12
cf/Documents/ИА_ФинансовоеПланирование/Forms/ФормаДокументаУпр/Ext/Form/Module.bsl		**93**	6	1	22	64
cf/Ext/SessionModule.bsl		**57**	14	24	19	-
cf/CommonModules/ИА_ФП/Ext/Module.bsl		**48**	1	3	43	1
cf/DataProcessors/ИА_КСУ_РабочийСтол/Forms/Форма_8_5_3/Ext/Form/Module.bsl		**41**	-	1	32	8
cf/Documents/ИА_ФинансовоеПланирование/Forms/Удалить_ФормаДокументаУпр_Типовая/Ext/Form/Module.bsl		**41**	8	1	22	10
cf/Tasks/ИА_ЗадачаПланаДействий/Forms/ФормаЗадачи/Ext/Form/Module.bsl		**40**	1	-	30	9
Другие		**2018**	179	287	1113	439

Top 10 модулей с ошибками.

Расчетное время на исправление всех ошибок 21 356 минут.

Правило	Технический долг (минут)	%	Количество замечаний	%
Конструкция "Попытка...Исключение...КонецПопытки" не содержит кода в исключении	4 140	19%	276	9%
Конструктор элемента стиля	2 605	12%	521	18%
Использование устаревшего метода "ТекущаяДата"	2 315	11%	463	16%
Использование объектов недоступных в Unix системах	1 860	9%	62	2%
Использование метода ПолучитьФорму	1 650	8%	110	4%
Нарушение парности использования методов "НачатьТранзакцию()" и "ЗафиксироватьТранзакцию()" / "ОтменитьТранзакцию()"	1 575	7%	105	4%
Выполнение запроса в цикле	1 340	6%	67	2%
Функция должна содержать возврат	780	4%	78	3%
Серверный экспортный метод формы	610	3%	122	4%
Другие	4 481	21%	1152	39%
Итоги	**21 356**	**100%**	**2956**	**100%**

TOP 10 правил, замечания по которым сформировали наибольший технический долг

Примеры ошибок

Нарушение парности использования методов "НачатьТранзакцию()" и "ЗафиксироватьТранзакцию()" / "ОтменитьТранзакцию()"

Начало транзакции и ее фиксация (отмена) должны происходить в контексте одного метода.

Примеры

Правильно

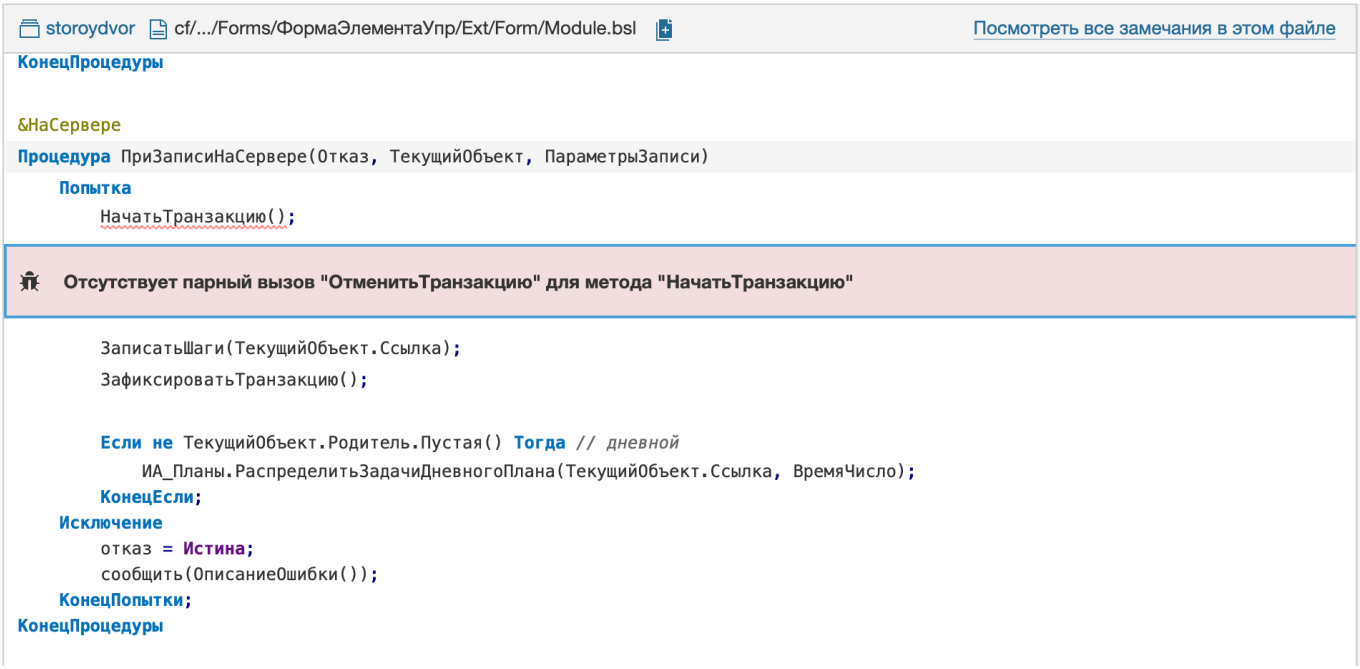
```
Процедура ЗаписатьДанныеВИБ()

    НачатьТранзакцию();

    Попытка
        ... // чтение или запись данных
        ДокументОбъект.Записать()
        ЗафиксироватьТранзакцию();
    Исключение
        ОтменитьТранзакцию();
        ... // дополнительные действия по обработке исключения
    КонецПопытки;

КонецПроцедуры
```

Неправильно

```
Процедура ЗаписатьДанныеВИБ()

    НачатьТранзакцию();
    ЗаписатьДокумент();

КонецПроцедуры;

Процедура ЗаписатьДокумент()

    Попытка
        ... // чтение или запись данных
        ДокументОбъект.Записать()
        ЗафиксироватьТранзакцию();
    Исключение
        ОтменитьТранзакцию();
    ... // дополнительные действия по обработке исключения
    КонецПопытки;

КонецПроцедуры
```

Источник: Транзакции: правила использования

Модуль: cf/Catalogs/ИА_ПланыДействий/Forms/ФормаЭлементаУпр/Ext/Form/Module.bsl

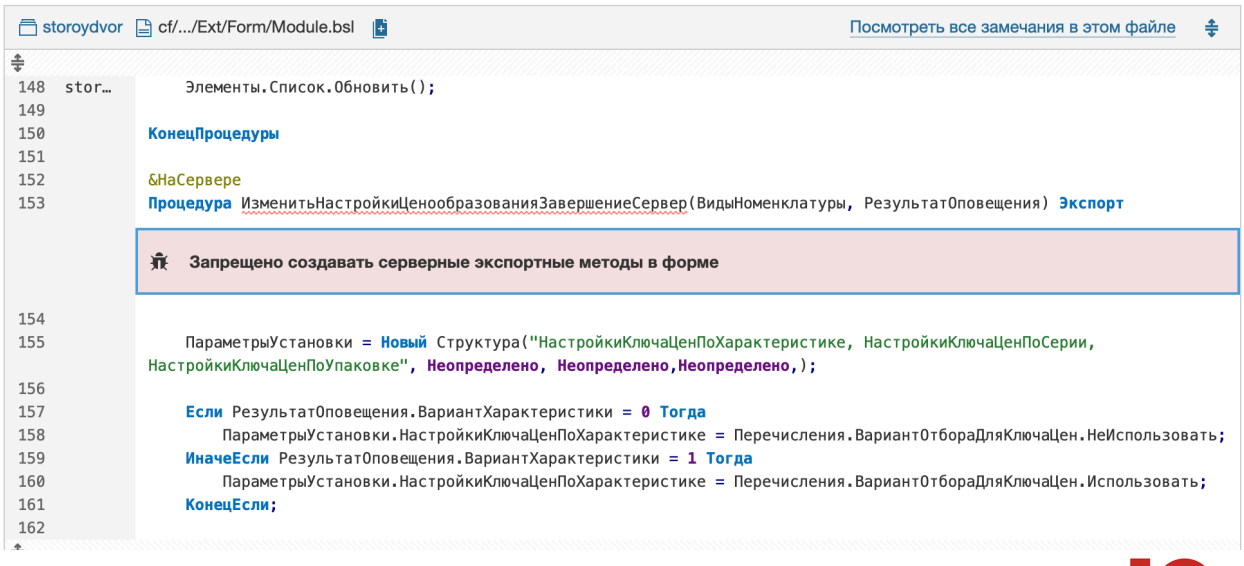


Серверный экспортный метод формы

В модуле формы можно объявлять экспортные методы, доступные в клиентском контексте (обычно это методы-обработчики событий оповещения формы). У экспортных методов формы может быть указана только директива компиляции НаКлиенте, так как для остальных практического смысла нет: обращение к методам формы из вне доступно только после вызова метода ПолучитьФорму, который доступен только на клиенте.
Указание экспортному методу формы иной директивы компиляции либо ее опускание считается ошибкой.
В некоторых версиях платформы 1С:Предприятие существовала ошибка, позволяющая использовать экспортные серверные методы форм, но проектировать прикладное решение с эксплуатированием ошибок платформы недопустимо.

Источник: Разработка интерфейса прикладных решений на платформе "1С:Предприятие". Глава 3.5. Исполнение модуля формы на клиенте и на сервере

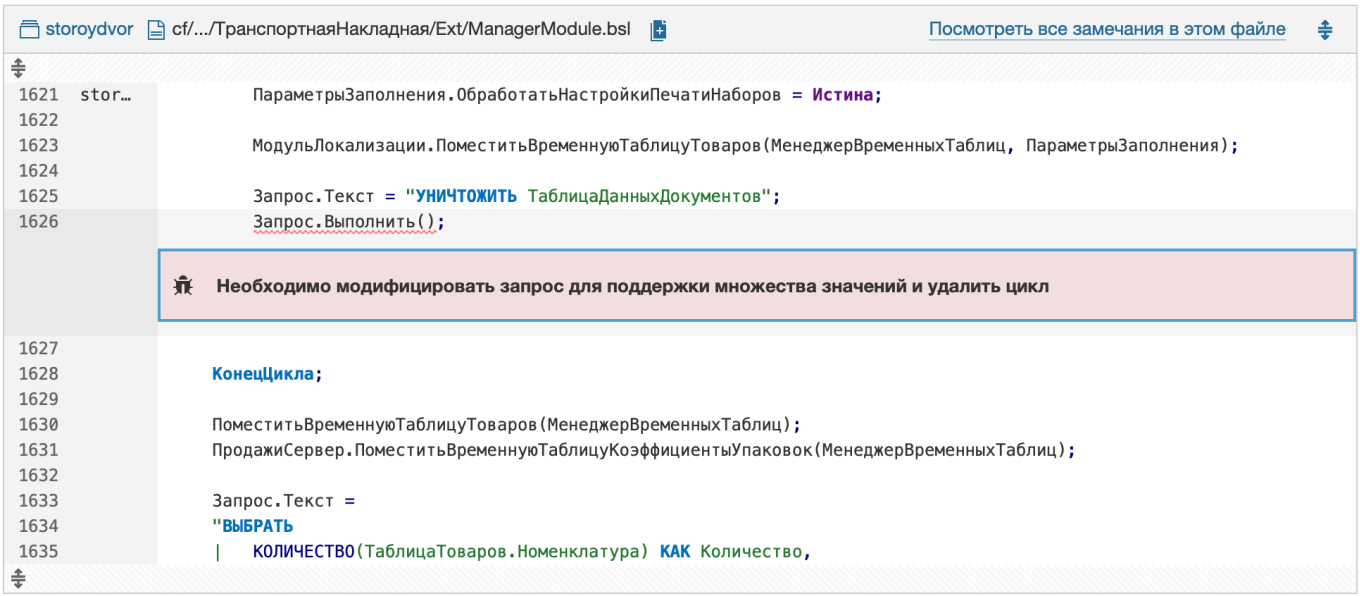
Модуль:
cf/Catalogs/ВидыНоменклатуры/Forms/ФормаСпискаДляНастройкиЦенообразования/Ext/Form/Module.bsl



Выполнение запроса в цикле

Исполнение запроса в цикле создает лишнюю нагрузку на сервер и увеличивает время выполнения процедуры.

Модуль: cf/Documents/ТранспортнаяНакладная/Ext/ManagerModule.bsl



Обзор уязвимостей

Уязвимости указывают на возможные проблемы с безопасностью.

Модуль	Важность	Всего	CRITICAL	MAJOR
cf/CommonModules/ОбменДаннымиСервер/Ext/Module.bsl		58	-	58
cf/CommonModules/ВзаиморасчетыСервер/Ext/Module.bsl		22	-	22
cf/Documents/ТранспортнаяНакладная/Ext/ManagerModule.bsl		17	-	17
cf/CommonModules/СкидкиНаценкиСервер/Ext/Module.bsl		12	-	12
cf/Documents/СчетФактураВыданный/Ext/ObjectModule.bsl		9	-	9
cf/CommonModules/ИА_БФС/Ext/Module.bsl		8	8	-
cf/Documents/СчетФактураВыданный/Ext/ManagerModule.bsl		8	-	8
cf/Catalogs/Пользователи/Forms/ФормаЭлемента/Ext/Form/Module.bsl		7	-	7
cf/Documents/КорректировкаРеализации/Ext/ManagerModule.bsl		7	-	7
Другие		260	62	198

Top 10 модулей с уязвимостями.

Правило	Технический долг (минут)	%	Количество замечаний	%
Итоги	**504**	**100%**	**70**	**100%**
Выполнение произвольного кода в общем модуле на сервере	345	68%	23	33%
Хранение ip-адресов в коде	45	9%	3	4%
Выполнение произвольного кода на сервере	39	8%	39	56%
Использование метода ПользователиОС	30	6%	2	3%
Хранение конфиденциальной информации в коде	30	6%	2	3%
Использование возможностей выполнения внешнего кода	15	3%	1	1%

TOP 10 правил, замечания по которым сформировали наибольший технический долг

Примеры уязвимостей

Выполнение произвольного кода на сервере

При разработке решений следует учитывать, что опасно использование не только непосредственного выполнения кода, написанного в режиме Предприятие, но и алгоритмов, где методами **Выполнить** или **Вычислить** исполняется код в серверных функциях и процедурах. Запрещено использование методов Выполнить и Вычислить в серверных методах модулей форм, команд, объектов и т.д.
Источник: Ограничения на использование Выполнить и Вычислить на сервере

Модуль: cf/DataProcessors/РегистрацияИзмененийДляОбменаДанными/Ext/ObjectModule.bsl

storoydvor cf/.../Ext/ObjectModule.bsl Посмотреть все замечания в этом файле

```
1993  stor…     Если Метаданные.ОбщиеМодули.Найти("ОбменДаннымиСобытия") = Неопределено Тогда
1994                Возврат Неопределено;
1995            КонецЕсли;
1996
1997            // Вызов ВычислитьВБезопасномРежиме не требуется, т.к. для вычисления передается строковый литерал.
1998            Возврат Вычислить("ОбменДаннымиСобытия");
                Запрещено выполнение произвольного кода на сервере
1999        КонецФункции
2000
2001        // Возвращает общий модуль СтандартныеПодсистемыСервер или Неопределено, если такого нет в составе конфигурации.
2002        //
2003        Функция ОбщийМодульСтандартныеПодсистемыСервер()
2004            Если Метаданные.ОбщиеМодули.Найти("СтандартныеПодсистемыСервер") = Неопределено Тогда
2005                Возврат Неопределено;
2006            КонецЕсли;
2007
```

Хранение конфиденциальной информации в коде

Запрещено хранить конфиденциальную информацию в коде. Список конфиденциальной информации:

- Пароли
- Персональные ключи доступа

Если в проекте 1С используется подсистема БСП, то хранение паролей можно организовать через безопасное хранилище.

Источник: Стандарт: Безопасное хранение паролей

Модуль: cf/CommonForms/КСУ_Новости/Ext/Form/Module.bsl

```
47  stor…
48
49          АдресСервера = Константы.КСУ_АдресСервераНовостей.Получить();
50          текПольз = ПараметрыСеанса.ТекущийПользователь;
51           Попытка
52              Соединение = Новый HTTPСоединение(АдресСервера,,"KSU","BycfqnPfobnf");
```

Используется хранение конфиденциальной информации в коде

```
53          Исключение
54              сообщить(ОписаниеОшибки());
55              возврат;
56          КонецПопытки;
57
58          ЗаписьJSON = Новый ЗаписьJSON;
59          ЗаписьJSON.УстановитьСтроку();
60          ЗаписатьJSON(ЗаписьJSON, новый
структура("Company,Text,User",имяОрг,Сообщение,строка(ПараметрыСеанса.ТекущийПользователь)));
61          СтрокаДляЗапроса = ЗаписьJSON.Закрыть();
```

Пропущен постфикс "ПолныеПрава"

Модули, выполняющиеся в привилегированном режиме, имеющие признак Привилегированный, должны именоваться с постфиксом "ПолныеПрава" (англ. "FullAccess"). Нарушение данного стандарта ведет к потенциальной уязвимости.

Например: РаботаСФайламиПолныеПрава, FilesFullAccess

Стандарт: Тексты модулей

Модуль: cf/CommonModules/КСУ_СтатистикиВеб/Ext/Module.bsl

Обзор дефектов кода

Дефекты кода, это группа замечаний, куда входит все что усложняет восприятие кода.

Модуль	Важность	Всего	CRITICAL	MAJOR	MINOR	INFO
cf/CommonModules/РасчетСебестоимостиНДС/Ext/Module.bsl		2 919	12	2 125	427	355
cf/Documents/ИА_ФинансовоеПланирование/Forms/ФормаДокументаУпр/Ext/Form/Module.bsl		2 863	51	138	408	2 266
cf/CommonModules/ВзаиморасчетыСервер/Ext/Module.bsl		1 839	102	364	739	634
cf/CommonModules/ИА_Статистики/Ext/Module.bsl		1 728	43	164	525	996
cf/DataProcessors/ИА_КСУ_РабочийСтол/Forms/Форма_8_5_3/Ext/Form/Module.bsl		1 716	26	134	806	750
cf/DataProcessors/ОргСхема/Forms/ФормаУпр/Ext/Form/Module.bsl		1 551	59	79	619	794
cf/Documents/ИА_ФП_Настройка/Forms/ФормаДокумента/Ext/Form/Module.bsl		1 507	34	21	225	1 227
cf/Catalogs/ОргСхема/Forms/ФормаРедактирования/Ext/Form/Module.bsl		1 260	18	73	499	670
cf/Tasks/ИА_ЗадачаПланаДействий/Forms/ФормаЗадачи/Ext/Form/Module.bsl		1 146	27	76	698	345

Top 10 модулей с дефектами.

Правило	Технический долг (минут)	%	Количество замечаний	%
Когнитивная сложность	16 980	9%	1132	1%
Пропущены пробелы слева или справа от операторов `+ - * / = % < > <> <= >=`, от ключевых слов, а так же справа от `,` и `;`	16 978	9%	16978	19%
Управляющие конструкции не должны быть вложены слишком глубоко	16 560	9%	552	1%
Использование логического "ИЛИ" в секции "ГДЕ" запроса	14 865	8%	991	1%
Использование синтаксической конструкции Если...Тогда...ИначеЕсли...	13 850	7%	1385	2%
Ограничение на длину строки	12 334	6%	12334	14%
Цикломатическая сложность	11 800	6%	472	1%
Каноническое написание ключевых слов	10 573	5%	10573	12%
Ограничение на размер метода	9 030	5%	301	0%
Другие	70 372	36%	43312	49%
Итоги	**193 342**	**100%**	**88030**	**100%**

TOP 10 правил, замечания по которым сформировали наибольший технический долг

Когнитивная сложность

Когнитивная сложность показывает на сколько сложно воспринимать написанный код.
Высокая когнитивная сложность явно указывает на необходимость проведения рефакторинга кода для облегчения его будущей поддержки.
Наиболее эффективным способом снижения когнитивной сложности является декомпозиция кода, дробление методов на более простые, а также оптимизация логических выражений.

Подсчет Когнитивной сложности
Ниже приведены правила анализа кода, условия повышения когнитивной сложности.

Каждый следующий блок увеличивает сложность на 1

```
// Цикл `Для каждого`
Для каждого Элемент Из Коллекция Цикл                // +1
КонецЦикла;

// Цикл `Для`
Для Ит = Начало По Конец Цикл                        // +1
КонецЦикла;

// Цикл `Пока`
Пока Условие Цикл                                    // +1
```

```
КонецЦикла;


// Условие
Если Условие Тогда                                  // +1

// Альтернативная ветвь условия
ИначеЕсли Условие2 Тогда                            // +1

// Ветвь по-умолчанию
Иначе
КонецЕсли;

// Тернарный оператор
Значение = ?(Условие, ЗначениеИстина, ЗначениеЛожь); // +1

Попытка
// Обработка исключения
Исключение                                          // +1
КонецПопытки;

// Переход на метку
Перейти ~Метка;                                     // +1

// Бинарные логические операции

Пока Условие ИЛИ Условие2 Цикл                      // +2
КонецЦикла;

Если Условие И Условие2 Тогда                       // +2

ИначеЕсли Условие2                                  // +1
        ИЛИ Условие3 И Условие4 Тогда               // +2

КонецЕсли;

Значение = ?(Условие ИЛИ Условие2 ИЛИ НЕ Условие3,  // +3
               ЗначениеИстина, ЗначениеЛожь);

Значение = Одно ИЛИ Второе;                         // +1

Значение = А <> В;                                  // +1
```

За каждый уровень вложенности, следующие блоки получают дополнительную единицу сложности

```
// Цикл `Для каждого`
Для каждого Элемент Из Коллекция Цикл
КонецЦикла;

// Цикл `Для`
```

```
Для Ит = Начало По Конец Цикл
КонецЦикла;

// Цикл `Пока`
Пока Условие Цикл
КонецЦикла;


// Условие
Если Условие Тогда
КонецЕсли;

// Тернарный оператор
Значение = ?(Условие, ЗначениеИстина, ЗначениеЛожь);

Попытка
// Обработка исключения
Исключение
КонецПопытки;

~Метка:
```

Альтернативные ветки, бинарные операции и переход на метку не увеличивают когнитивную сложность при вложении

Примеры

Ниже на примерах кода произведен рассчет когнитивной сложности методов.

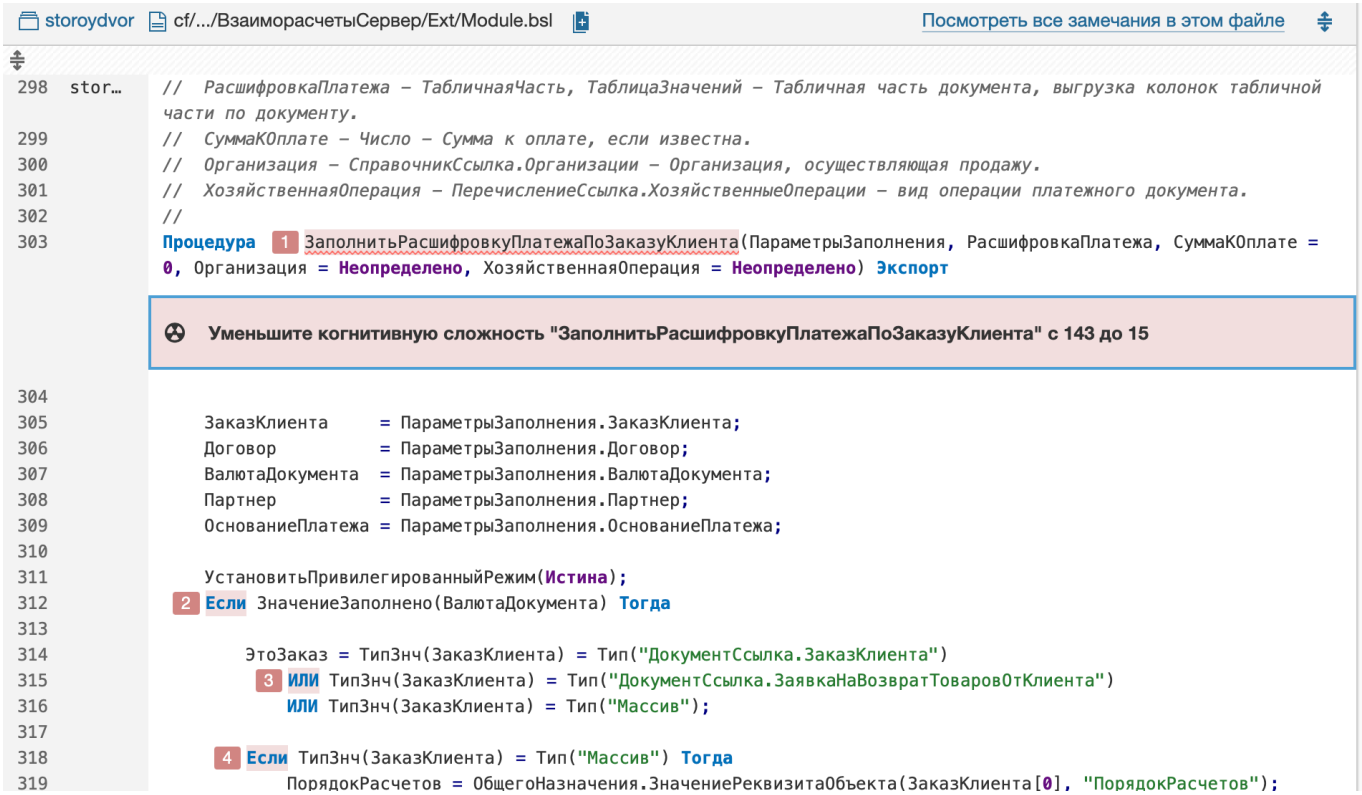
```
Функция Пример1(ТипКласса)
    Если ТипКласса.Неизвестен() Тогда                                          // +1, условие,
вложенности нет
        Возврат Символы.НеизвестныйСимвол;
    КонецЕсли;

    НеизвестностьНайдена = Ложь;
    СписокСимволов = ТипКласса.ПолучитьСимвол().Потомки.Поиск("имя");
    Для Каждого Символ Из СписокСимволов Цикл                                  // +1, цикл,
вложенности нет
        Если Символ.ИмеетТип(Символы.Странное)                                 // +2, условие
вложенное в цикл, вложенность 1
            И НЕ Символы.Экспортный() Тогда                                    // +1,
логическая операция, вложенность не учитывается

            Если МожноПереопределить(Символ) Тогда                             // +3,
вложенное условие, вложенность 2
                Переопредялемость = ПроверитьПереопределяемость(Символ, ТипКласса);
                Если Переопределяемость = Неопределено Тогда                   // +4,
вложенное условие, вложенность 3
                    Если НЕ НеизвестностьНайдена Тогда                         // +5,
вложенное условие, вложенность 4
                        НеизвестностьНайдена = Истина;
```

```
                    КонецЕсли;
                ИначеЕсли Переопределяемость Тогда                                  // +1,
альтернативная ветвь условия, вложенность не учитывается
                    Возврат Символ;
                КонецЕсли;
            Иначе                                                                   // +1, ветвь
по-умолчанию, вложенность не учитывается
                Продолжить;
            КонецЕсли;
        КонецЕсли;
    КонецЦикла;

    Если НеизвестностьНайдена Тогда                                                 // +1,
вложенности нет
        Возврат Символы.НеизвестныйСимвол;
    КонецЕсли;

    Возврат Неопределено;
КонецФункции

Функция Пример2(Документ)
    НачатьТранзакцию();
    НадоПровести = ?(Документ.Проведен, ЛОЖЬ,
// +1, тернарный оператор
                                        ?(Документ.ПометкаУдаления, ЛОЖЬ, ИСТИНА));
// +2, вложенный тернарный оператор, вложенность 1
    Попытка
// +0, попытка, повышает уровень вложенности
        ДокументОбъект = Документ.ПолучитьОбъект();
        Если ДокументОбъект.Проведен Тогда
// +2, вложенное условие, вложенность 1
            Для Каждого СтрокаТабличнойЧасти Из ДокументОбъект.ТабличнаяЧасть Цикл
// +3, вложенный цикл, вложенность 2
                Если СтрокаТабличнойЧасти.Колонка1 = 7
// +4, вложенное условие, вложенность 3
                        ИЛИ СтрокаТабличнойЧасти.Колонка2 = 7 Тогда
// +1, логическая операция, вложенность не учитывается
                    Продолжить;
                КонецЕсли;
                Если СтрокаТабличнойЧасти.Колонка4 > 1 Тогда
// +4, вложенное условие, вложенность 3
                    Прервать;
                Иначе
// +1, ветвь по-умолчанию, вложенность не учитывается
                    Если СтрокаТабличнойЧасти.Колонка1 + СтрокаТабличнойЧасти.Колонка2 = 2 Тогда
// +5, вложенное условие, вложенность 4
                        СтрокаТабличнойЧасти.Колонка10 = СтрокаТабличнойЧасти.Колонка1 * 2;
                    КонецЕсли;
                КонецЕсли;
            КонецЦикла;
        Иначе
// +1, ветвь по-умолчанию, вложенность не учитывается
            НадоПровести = ДокументОбъект.Дата > ТекущаяДата();
// +1, логическая операция, вложенность не учитывается
```

```
            Перейти ~Метка;
// +1, переход на метку, вложенность не учитывается
        КонецЕсли;

        Если НадоПровести Тогда
// +2, вложенное условие, вложенность 1
            ДокументОбъект.Записать(РежимЗаписиДокумента.Проведение);
        ИначеЕсли НЕ НадоПровести Тогда
// +1, альтернативная ветвь, вложенность не учитывается
            ДокументОбъект.Записать(РежимЗаписиДокумента.Запись);
        Иначе
// +1, ветвь по-умолчанию, вложенность не учитывается
            ВызватьИсключение "Как так-то?";
        КонецЕсли;
    Исключение
// +1, обработка исключения
        ПовторнаяЗапись = ЛОЖЬ;
        Попытка
// +0, попытка, повышает уровень вложенности
            Если ДокументОбъект.Проведен Тогда
// +3, вложенное условие, вложенность 2
                ДокументОбъект.Записать(РежимЗаписиДокумента.Запись);
            КонецЕсли;
        Исключение
// +2, обработка исключения, вложенность 1
            ПовторнаяЗапись = ИСТИНА;
        КонецПопытки;
        Если Не ПовторнаяЗапись Тогда
// +2, вложенное условие, вложенность 1
            Пока ТранзакцияАктина() Цикл
// +3, вложенный цикл, вложенность 2
                ОтменитьТранзакцию();
            КонецЦикла;
        КонецЕсли;
        ВызватьИсключение "Ошибка"
    КонецПопытки;

    ~Метка:
    Возврат Неопределено;
КонецФункции
```

Источник: [Cognitive complexity, ver. 1.4]

Модуль: cf/CommonModules/ВзаиморасчетыСервер/Ext/Module.bsl



Цикломатическая сложность

Цикломатическая сложность программного кода является одной из наиболее старых метрик, впервые она была упомянута в 1976 году Томасом МакКэбом.
Цикломатическая сложность показывает минимальное число необходимых тестов. Наиболее эффективным способом снижения цикломатической сложности является декомпозиция кода, дробление методов на более простые, а также оптимизация логических выражений.

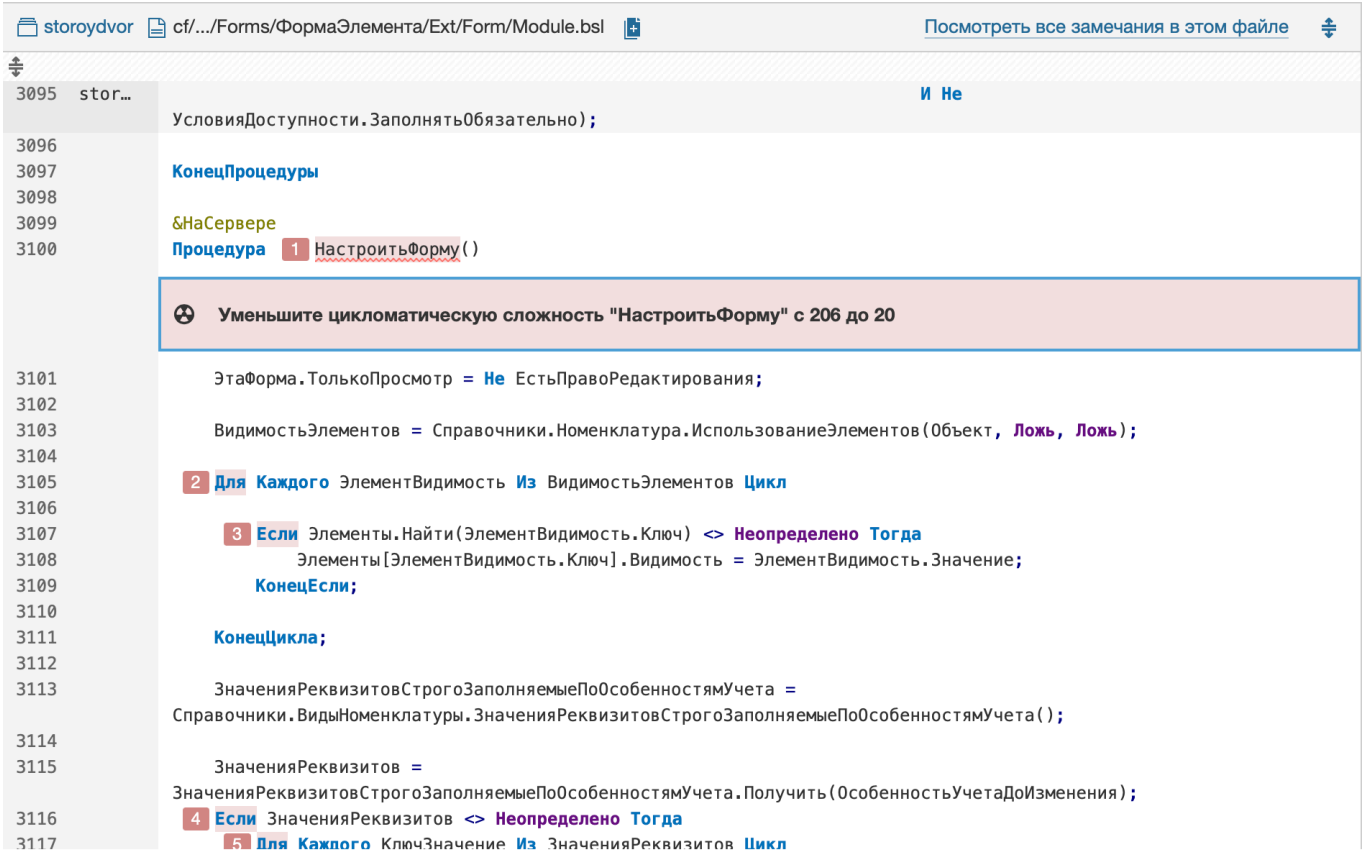
Цикломатическая сложность увеличивается на 1 за каждую конструкцию:

- `Для ... По .. Цикл`
- `Для каждого ... Из ... Цикл`
- `Если ... Тогда`
- `ИначеЕсли ... Тогда`
- `Иначе`
- `Попытка ... Исключение ... КонецПопытки`
- `Перейти ~Метка`
- Бинарные операции `И ... ИЛИ`
- Тернарный оператор
- `Процедура`
- `Функция`

Источники:

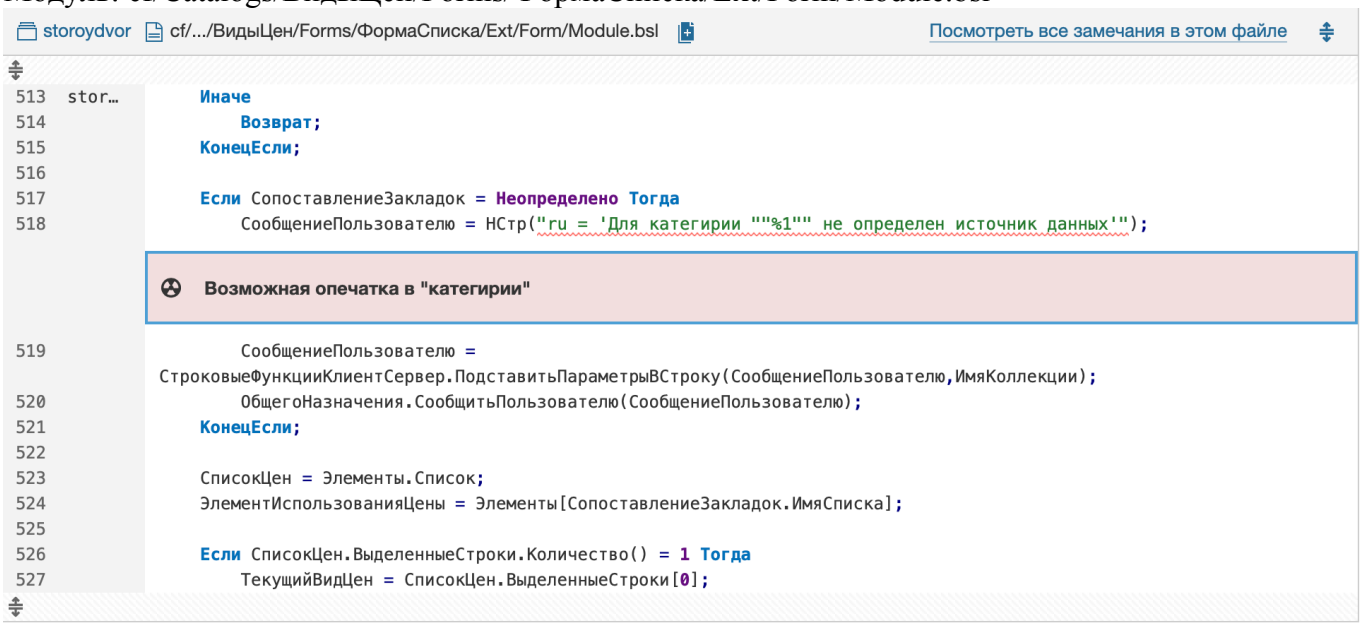
- Cyclomatic Complexity PHP
- Цикломатическая сложность

Модуль: cf/Catalogs/ВидыНоменклатуры/Forms/ФормаЭлемента/Ext/Form/Module.bsl



Опечатка

Модуль: cf/Catalogs/ВидыЦен/Forms/ФормаСписка/Ext/Form/Module.bsl

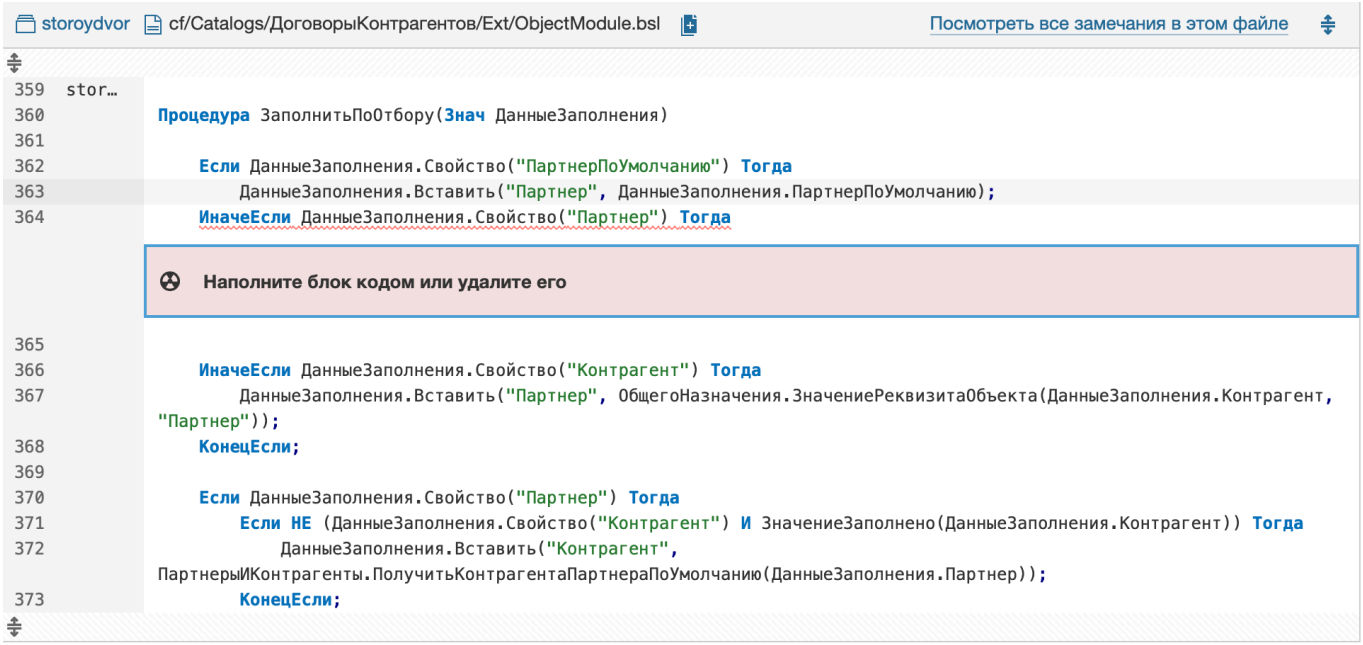


Пустой блок кода

Пустые блоки являются признаком возможной ошибки:

- Забыли реализовать
- Удалили содержимое

Пустые блоки кода должны быть наполнены либо удалены.
Модуль: cf/Catalogs/ДоговорыКонтрагентов/Ext/ObjectModule.bsl



Заключение

На данный момент конфигурация находится в хорошем состоянии, за исключением группы подсистем с префиксом КСУ.

В соответствии с концепцией чистого кода рекомендуем настроить процесс разработки с использованием хранилища конфигурации, проводить регулярную проверку нового кода во избежание накопления технического долга.

Внедрение хранилища конфигурации позволит упростить групповую разработку, а также обеспечит идентификацию автора кода.

Зачем нужно управлять качеством кода можно почитать по ссылкам:

1. https://infostart.ru/1c/articles/1096770/#_7ztptmi3s70k
2. https://infostart.ru/1c/articles/622617/
3. https://rarus.ru/publications/20210827-ot-ekspertov-kachestvennyj-kod-v-1c-avtomatizirovannaya-proverka-konfiguracij-sonarqube-492925/

Плюсы внедрения анализатора кода:

1. На проектах, где участвуют 3-5 и более разработчиков проверять код вручную невозможно. Оставлять код без проверки, означает копить технический долг, который в дальнейшем придется возвращать, т.е. оплачивать разработчикам время на его устранение.
2. Оперативное выявление замечаний позволяет избавиться от них с наименьшими потерями:
    a. Для внутренней команды разработки проще здесь и сейчас в новом коде устранить все замечания, чем спустя какое-то время разбираться со старым кодом
    b. Для разработки проектов на подряде, это возможность устранить все замечания в коде проекта за счет подрядчика, в оговоренный бюджет.
3. Ускорение процесса код ревью и как следствие ускорение установки доработок в рабочую базу.
4. Возможность использовать на проекте специалистов среднего уровня, без необходимости объяснения им правил соблюдения стандартов разработки (к каждой ошибке анализатор прикладывает подробное описание), достаточно предоставить им доступ на сервер SonarQube, и контролировать что выявленные ошибки исправляются.