ПРИЛОЖЕНИЕ К АУДИОКНИГЕ

Никита Зайцев aka WildHare

Путь 1С-разработки

не спеша, эффективно и правильно

(Приложение содержит таблицы и схемы из печатной версии книги)

ПИТЕР®

Санкт-Петербург · Москва · Минск

2024

Рис. 1.1. Памятка заказчику

Листинг 2.1. Решение элементарной задачи в китайском стиле

```
НомерСтрокой = "";
Если ОчереднойНомер < 9 Тогда
    НомерСтрокой = "00000" + СокрЛП(ОчереднойНомер);
ИначеЕсли ОчереднойНомер < 99 Тогда
    НомерСтрокой = "0000" + СокрЛП(Формат(ОчереднойНомер, "ЧГ=0"));
ИначеЕсли ОчереднойНомер < 999 Тогда
    НомерСтрокой = "000" + СокрЛП(Формат(ОчереднойНомер, "ЧГ=0"));
ИначеЕсли ОчереднойНомер < 9999 Тогда
    НомерСтрокой = "00" + СокрЛП(Формат(ОчереднойНомер, "ЧГ=0"));
ИначеЕсли ОчереднойНомер < 99999 Тогда
    НомерСтрокой = "0" + СокрЛП(Формат(ОчереднойНомер, "ЧГ=0"));
ИначеЕсли ОчереднойНомер < 999999 Тогда
    НомерСтрокой = СокрЛП(Формат(ОчереднойНомер, "ЧГ=0"));
КонецЕсли;
```

Листинг 2.2. Решение элементарной задачи в инженерном стиле

```
НомерСтрокой = Строка(Формат(ОчереднойНомер, "ЧГ=0"));
ДлинаНомера = СтрДлина(НомерСтрокой);
Если ДлинаНомера < 6 Тогда
    Для Счетчик = 1 По 6 - ДлинаНомера Цикл
        НомерСтрокой = "0" + НомерСтрокой;
    КонецЦикла;
КонецЕсли;
```

Листинг 2.3. Решение элементарной задачи в экспертном стиле

```
НомерСтрокой = Формат(ОчереднойНомер, "ЧЦ=6; ЧДЦ=0; ЧВН=; ЧГ=0");
```

Листинг 2.4. Проверка на элементарный тип в индусском стиле

```
КодНекорректногоТипаДанныхЧисло   = -1111;
Попытка
    ТипИсходныхДанных = ТипЗнч(ВходящееЧисло);
    Если ВходящееЧисло = Null Тогда
        Возврат Число(0);
    ИначеЕсли ТипИсходныхДанных = Тип("Число") Тогда
        Возврат Число(ВходящееЧисло);
    Иначе
        Возврат КодНекорректногоТипаДанныхЧисло;
    КонецЕсли;
Исключение
    Возврат КодНекорректногоТипаДанныхЧисло;
КонецПопытки;
```

Листинг 2.5. Шизокодированное сообщение пользователю

```
Если УспешноеВыполнение И ЕстьОшибки = Ложь Тогда
    Предупреждение(«Обработка успешно выполнена.»);
ИначеЕсли УспешноеВыполнение И ЕстьОшибки = Истина Тогда
    Предупреждение(«Внимание! Обработка была выполнена с ошибками»);
Иначе
    Предупреждение(«При выполнении обработки возникли ошибки!»);
КонецЕсли;
```

Листинг 2.6. Шизокодированное копирование реквизитов

```
ИмяФайла = "bank.dbf";
БанковскийСчет = БанковскийСчетВыгрузки.БанковскийСчет;
КаталогФайловВыгрузки = БанковскийСчетВыгрузки.ИмяКаталогаДанных;
КаталогАрхиваФайловВыгрузки = БанковскийСчетВыгрузки.ИмяКаталогаАрхива;
КаталогИмпортаФайловВыгрузки = БанковскийСчетВыгрузки.ИмяКаталогаИмпорта;
НомерРС = БанковскийСчетВыгрузки.БанковскийСчет.НомерСчета;
МаксимальноеКоличество = БанковскийСчетВыгрузки.КоличествоПлатежек;
```

Листинг 2.7. Шизокодированный выбор вида операции

```
Если ЭтаФорма.ВариантПечати = 1 Тогда
    СвойстваПринтера = ПолучитьПараметрыПринтера(Принтер);
    Если Не СвойстваПринтера.ПринтерИспользуется Тогда
        Сообщить(«Использование указанного принтера запрещено.»);
    КонецЕсли;
КонецЕсли;
ОбъектыПечати = ЗаполнениеМассиваДокументовДляПечати();
Если ЭтаФорма.ВариантПечати = 0 Тогда
    ПараметрыПечати = ПечатьСервер.СтруктураПП(ОбъектыПечати,
                                               "МассоваяПечатьПросмотр");
Иначе
    ПараметрыПечати = ПечатьСервер.СтруктураПП(ОбъектыПечати,
                                               "МассоваяПечатьНаПринтер");
    ПараметрыПечати.ИмяПринтера = СвойстваПринтера.ИмяПринтера;
КонецЕсли;
```

Листинг 2.8. Простое пояснение к непростому алгоритму

```
// tasks by workers, Round Robin
```

Листинг 2.9. Избыточный комментарий в стиле «пересказ кода»

```
// Обходим коллекцию в цикле, добавляем строку в табличную часть,
// заполняем на основе коллекции.
Для Каждого Элемент Из Коллекция Цикл
    НоваяСтрока = ТабличнаяЧасть.Добавить();
    ЗаполнитьЗначенияСвойств(НоваяСтрока, Элемент);
КонецЦикла;
```

Листинг 2.10. Абсолютно бесполезный технический заголовок метода

```
// Процедура запускает обработку очереди заданий.
//
// Параметры:
//    Настройки - Структура - Настройки процедуры.
//
Процедура ЗапуститьОбработкуОчередиЗаданий(Настройки, БыстрыйРежим =
```

ГЛАВА 3. ПРОРАБАТЫВАЕМ ПРАКТИКУ

Листинг 3.1. Неудобочитаемый интерфейс метода с множеством параметров

```
Процедура СоздатьДокументыПоСхеме(Отказ, СхемаТО, ПараметрыТО,
                                  ПакетОпераций, ЭлементПакета,
                                  УправляющаяВыборка = Неопределено,
                                  РасшифровкаВыборки = Неопределено,
                                  СписокИсключений = Неопределено,
                                  СтруктураДопДанных = Неопределено,
                                  ЭтоФоноваяЗадача = Ложь,
                                  ПоказыватьСостояние = Ложь,
                                  ОтключитьСписокИсключений = Ложь) Экспорт
```

Листинг 3.2. Крайняя степень пренебрежения правилами структурирования кода

```
Процедура МагическийПоискДоговора_2011(
    Отказ,
    СтрокаСообщения,
    ДатаЗагрузки,
    Район_СГУ,
    ТаблицаТаблицПоДатам,
    КонтрагентыДляОбычныхРайоновСГУ,
    АналитикаКонтрагента,
    АналитикаПоставщика, // Для проверки флага "Не выгружать в КПУ"
    ФлагРазделенияУслугНаДваДокумента,
    ДоговорГУЖАДляДеления,
    СубсчетДляДеления,
    ЛицевойСчетГУЖА,
    АналитическийТипДоговораДляДеления,
    ДокументПоступленияПлатежейНачисления,
    РежимТранзакции = Ложь,
    ВключеноОтслеживание = Ложь,
    ВключеноПротоколирование = Ложь,
    УказательПротокола = Неопределено,
    ЭтоФоноваяЗадача = Ложь,
    Идентификатор = "",
    Индикатор = Неопределено,
    БылаОшибка) Экспорт
```

Листинг 3.3. Правильная техника работы с параметрами

```
// Корневая процедура-контейнер для последовательного вызова
// бизнес-логики определенной подсистемы.
Процедура СделатьВсеКакНадо(ЭтоФоновыйРежим = Ложь, Отказ = Ложь) Экспорт
    // Бизнес-логика, определяющая основные параметры вызова нашего метода,
    // опущена. Предполагается, что они уже вычислены
    // (например, мы находимся внутри некоего цикла).
    ПараметрыМетода = Новый Структура;
    ПараметрыМетода.Вставить("ФоновыйРежим", ЭтоФоновыйРежим);
    // Значения других дополнительных параметров будут взяты по умолчанию.
    ЗагрузитьРасчетныеДокументы(ВнешняяСистема, РасчетныйПериод,
                                Организация, ПараметрыМетода);
    // Продолжение бизнес-логики корневой процедуры.
КонецПроцедуры // СделатьВсеКакНадо()

// Загружает расчетные данные из внешней информационной системы
// через веб-сервис.
Процедура ЗагрузитьРасчетныеДокументы(ВнешняяСистема, РасчетныйПериод,
                    Организация, ПараметрыМетода = Неопределено) Экспорт
    ЗагрузитьРасчетныеДокументы_Проверить(ПараметрыМетода);
    // Параметры проверены, далее следует бизнес-логика метода.
КонецПроцедуры // ЗагрузитьРасчетныеДокументы()

// Проверяет и заполняет значениями по умолчанию структуру параметров
```

```
// метода ЗагрузитьРасчетныеДокументы().
Процедура ЗагрузитьРасчетныеДокументы_Проверить(ПараметрыМетода =
                                                    Неопределено)
    Если ПараметрыМетода = Неопределено   Тогда
        ПараметрыМетода = Новый Структура;
    КонецЕсли;
    СхемаМетода = Новый Структура;
    СхемаМетода.Вставить("РежимОтладки", Ложь);
    СхемаМетода.Вставить("РежимТранзакции", Истина);
    СхемаМетода.Вставить("ФоновыйРежим", Ложь);
    // Таких дополнительных параметров метода может быть
    // сколь угодно много и самых разных типов.
    // Для лаконичности примера здесь только три простых флага.
    Для Каждого ЭлементСхемы Из СхемаМетода Цикл
        Если Не ПараметрыМетода.Свойство(ЭлементСхемы.Ключ) Тогда
            ПараметрыМетода.Вставить(ЭлементСхемы.Ключ,
                                     ЭлементСхемы.Значение);
            // В промышленном решении здесь может быть расположен вызов
            // универсальной процедуры, которая не только дополнит
            // заданные параметры метода значениями по умолчанию, но и
            // проверит обязательность/валидность заданных параметров,
            // при необходимости сформирует текст ошибки и вызовет
            // исключение, ну и так далее. В учебном примере все эти
            // сложности отсутствуют, показана только сама идея.
        КонецЕсли;
    КонецЦикла;
КонецПроцедуры // ЗагрузитьРасчетныеДокументы_Проверить()
```

Листинг 3.4. Аннотация курильщика – бесполезный пересказ названия

```
// Выполняет удаление лишних документов.
//
// Параметры:
//    Отказ - Булево, флаг указывает на неудачную попытку синхронизации.
//    ТаблицаИзОбработки – Не определено, если запускается из
//                         регламентного задания. Содержит таблицу
//                         значений НайденныеДокументы со списком
//                         удаляемых документов, если запускается из
//                         формы обработки.
//    ЭтоФоновоеЗадача - Булево, флаг указывает на запуск процедуры в
//                       качестве фоновой.
//    Пользователь - СправочникСсылка.Пользователи, пользователь, который
//                   инициировал запуск синхронизации.
//    РежимТранзакции - Булево, флаг указывает на необходимость
//                      использования режима управляемой транзакции.
//
Процедура УдалениеЛишнихДокументов(Отказ, ТаблицаИзОбработки,
                                   ЭтоФоновоеЗадача = Ложь,
                                   Пользователь = Неопределено ,
                                   РежимТранзакции = Неопределено,
                                   Режим2012 = Ложь,
                                   Протокол = Неопределено) Экспорт
```

Листинг 3.5. Аннотация здорового человека – краткая и осмысленная

```
// Извлекает из репозитория параметры указанных источников данных,
// формирует из них структуру (параметры, имена полей и параметров,
// необходимые временные таблицы и т. д.).
//
// Параметры:
//    ВхДанные - СправочникСсылка.УниверсальныеИсточникиДанных,Массив -
//               Источник (-и) данных, которые необходимо подготовить к
//               выполнению.
//
// Возвращаемое значение:
//    Структура - Полный контекст набора источников.
//
Функция КонтекстДинамическихДанных(ВхДанные) Экспорт
```

Листинг 3.6. Аннотация без аннотации – так тоже можно

```
// Аннотация не требуется, см. имя метода.
//
// Параметры:
//    Нет.
//
Процедура ОткатитьТранзакцию()
    Пока ТранзакцияАктивна() Цикл
        ОтменитьТранзакцию();
    КонецЦикла;
КонецПроцедуры // ОткатитьТранзакцию()
```

Листинг 3.7. Неупорядоченная куча литералов

```
ПараметрыОткрытия = Новый Структура("Ссылка,Комментарий,Наименование,
                                    |Ответственный,ПоУмолчанию,ПометкаУдаления,
                                    |НавигационнаяСсылкаНаУсловие,
                                    |Порядок,ГруппаТомов,ПредставлениеУсловия,
                                    |УникальныйИдентификаторФормыРодителя,
                                    |ЕстьПравилоПоУмолчанию");
```

Листинг 3.8. Упорядоченный по алфавиту массив литералов

```
ПараметрыОткрытия = Новый Структура("ГруппаТомов,ЕстьПравилоПоУмолчанию,
                                    |Комментарий,НавигационнаяСсылкаНаУсловие,
                                    |Наименование,Ответственный,ПометкаУдаления,
                                    |Порядок,ПоУмолчанию,ПредставлениеУсловия,
                                    |Ссылка,УникальныйИдентификаторФормыРодителя")
```

Листинг 3.9. Упражнение на мысленную компиляцию условия

```
ФлагДальнейшейОбработки = (Атрибуты.ФункциональныйТип = АгентскийТип
Или Атрибуты.ФункциональныйТип = ТипАгентскогоНФУ)
И Не ПриемлемыеСтатусы.Получить(ТекущиеСтатусы.Статус) = Неопределено
И ДатаПлатежей >= НачалоДня(Атрибуты.ДатаНачалаДействия)
И ДатаПлатежей <= КонецДня(?(Атрибуты.ДатаЗавершенияДействия =
'00010101', '29991231', Атрибуты.ДатаЗавершенияДействия))
И Не СписокОбрабатываемыхОбычныхРайонов.Получить(РайоныСГУ.Район_СГУ) =
Неопределено
И (РайоныСГУ.СтрокаДействует = Истина
Или РайоныСГУ.Приложение.Владелец = Атрибуты.ФункциональныйТип)
И (Не Атрибуты.ПометкаУдаления = Истина
Или АналитикиСторон.АналитикаДействующая = Истина);
```

Листинг 3.10. Ненужное уплотнение нескольких строк кода в одну

```
Возврат Запрос.Выполнить().Выгрузить().Свернуть("Ссылка").
ВыгрузитьКолонку("Ссылка");
```

Листинг 3.11. Размен количества строк на осмысленность и управляемость кода

```
// Предполагается, что запрос является пакетом запросов, и в целях отладки
// выборки из промежуточных временных таблиц помещаются
// в отдельные запросы пакета.
РезультатПакета = Запрос.ВыполнитьПакет();
РезультатЗапроса = РезультатПакета[РезультатПакета.Количество() - 1];
ТаблицаЗапроса = РезультатЗапроса.Выгрузить();
Ссылки = ТаблицаЗапроса.Свернуть("Ссылка").ВыгрузитьКолонку("Ссылка");
Возврат Ссылки;
```

Листинг 3.12. Еще одно упражнение на мысленную компиляцию кода

```
Если ((ТекДанные.ТипДоговора = ТипАгент
    И ТекДанные.ЭтоФорма2 = Истина
    И ПравильныеРСО.Получить(ТекДанные.РСО) <> Неопределено)
    Или (ТекДанные.ТипДоговора = ТипМодель
    И ТекДанные.ВидУслуги = Перечисления.ТипЖКУ.Жилищное
    И ТекДанные.ДатаЗавершения <= КонецМесяца(ДатаОбработки)))
    И ТекДанные.ДатаЗавершения <= КонецМесяца(ДатаОбработки)))
    И ТекДанные.Статус = Перечисления.СтатусДоговора.Действует
    И ПлохиеУК.Получить(ТекДанные.
ВтораяСторона) = Неопределено Тогда
```

Листинг 3.13. Дефрагментация сложного условия

```
// Согласно правилу первого экрана, этот блок в промышленном коде
// хорошо бы оформить отдельным методом.

Операнды = Новый Массив;

Операнды.Добавить(ТекДанные.ТипДоговора = ТипАгент);
Операнды.Добавить(ТекДанные.ЭтоФорма2 = Истина);
Операнды.Добавить(ПравильныеРСО.Получить(ТекДанные.РСО) <> Неопределено);
АгентыОК = Конъюнкция(Операнды);

Операнды.Очистить();

Операнды.Добавить(ТекДанные.ТипДоговора = ТипМодель);
Операнды.Добавить(ТекДанные.ВидУслуги = Перечисления.ТипЖКУ.Жилищное);
Операнды.Добавить(ТекДанные.ДатаЗавершения <= КонецМесяца(ДатаОбработки));
МодельОК = Конъюнкция(Операнды);

Операнды.Очистить();

Операнды.Добавить(ТекДанные.Статус = Перечисления.СтатусДоговора.Действует);
Операнды.Добавить(ПлохиеУК.Получить(ТекДанные.ВтораяСторона)= Неопределено);
ДоговорОК = Конъюнкция(Операнды);

ПроверкаПройдена = (АгентыОК Или МодельОК) И ДоговорОК;
```

Листинг 3.14. Универсальная логическая функция

```
// Сопоставляет массив входящих условий оператором И.
Функция Конъюнкция(Операнды) Экспорт
    Для Каждого Операнд Из Операнды Цикл
        Если Операнд <> Истина Тогда
            Возврат Ложь;
        КонецЕсли;
    КонецЦикла;
    Возврат Истина;
КонецФункции // Конъюнкция()
```

Листинг 3.15. Два полярных стиля оформления условий

```
// Утверждающий стиль оперирует понятиями "хорошо" и "можно".
МожноЗаписать = Накладная.Товары.Количество() > 0;
// Отрицающий стиль ставит на пути исполнения кода негативные блоки.
НельзяЗаписать = (Накладная.Товары.Количество() = 0
                  Или Накладная.Товары.Итог("Количество") = 0);
```

Листинг 3.16. Три типичные проблемы «условных» функций

```
// Здесь сразу три проблемы:
// - Функция поименована как процедура.
// - Проверить = Ложь — это значит "не удалось проверить"?
// - Третий (риторический) вопрос — а где описание ошибки?
Если Не Накладная.ПроверитьЗаполнение() Тогда
    // Здесь может быть, например, запись ошибки в ЖР.
    Продолжить;
КонецЕсли;
Накладная.Записать(РежимЗаписиДокумента.Проведение);
```

Листинг 3.17. И еще одно упражнение на мысленную компиляцию кода

```
// Какое именно условие здесь описано, читателю предлагается разобрать
// самостоятельно. Сон разума пост сдал.
Если (Не НеХватаетТовара И Не КонтрольНеОтключен
     Или НеХватаетТовара И КонтрольНеОтключен) =
     (Не ВсегдаПроверятьКоличество) Тогда
```

Листинг 3.18. Безалаберное использование исключения

```
Попытка
    Текст = СтроковыеФункцииКлиентСервер.ПодставитьПараметрыВСтроку(Шаблон,
                                                                    Ссылка);
Исключение
    Текст = СтрЗаменить(Шаблон,"%1", Ссылка);
КонецПопытки;
```

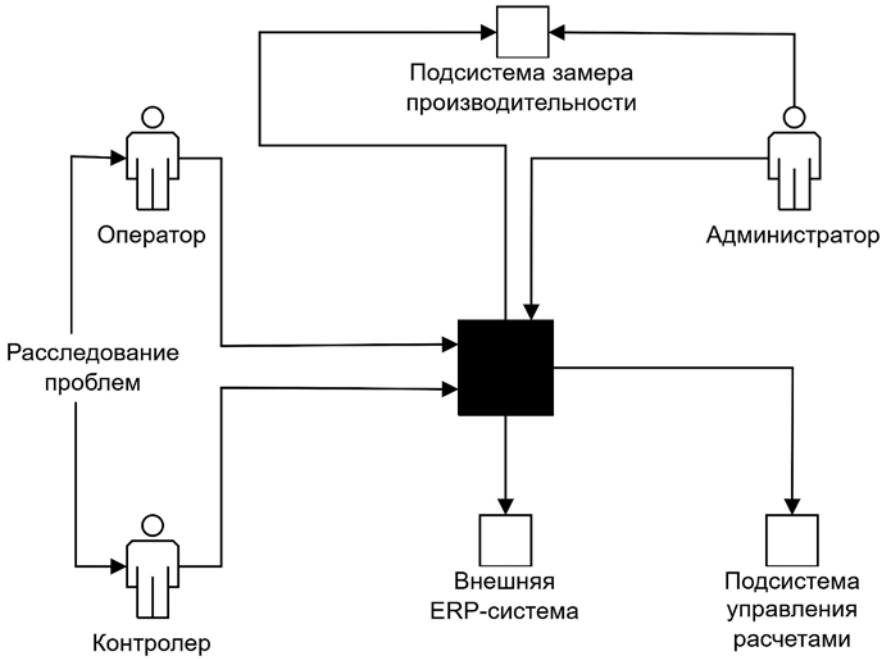

Рис. 3.1. Диаграмма геоцентрического черного ящика

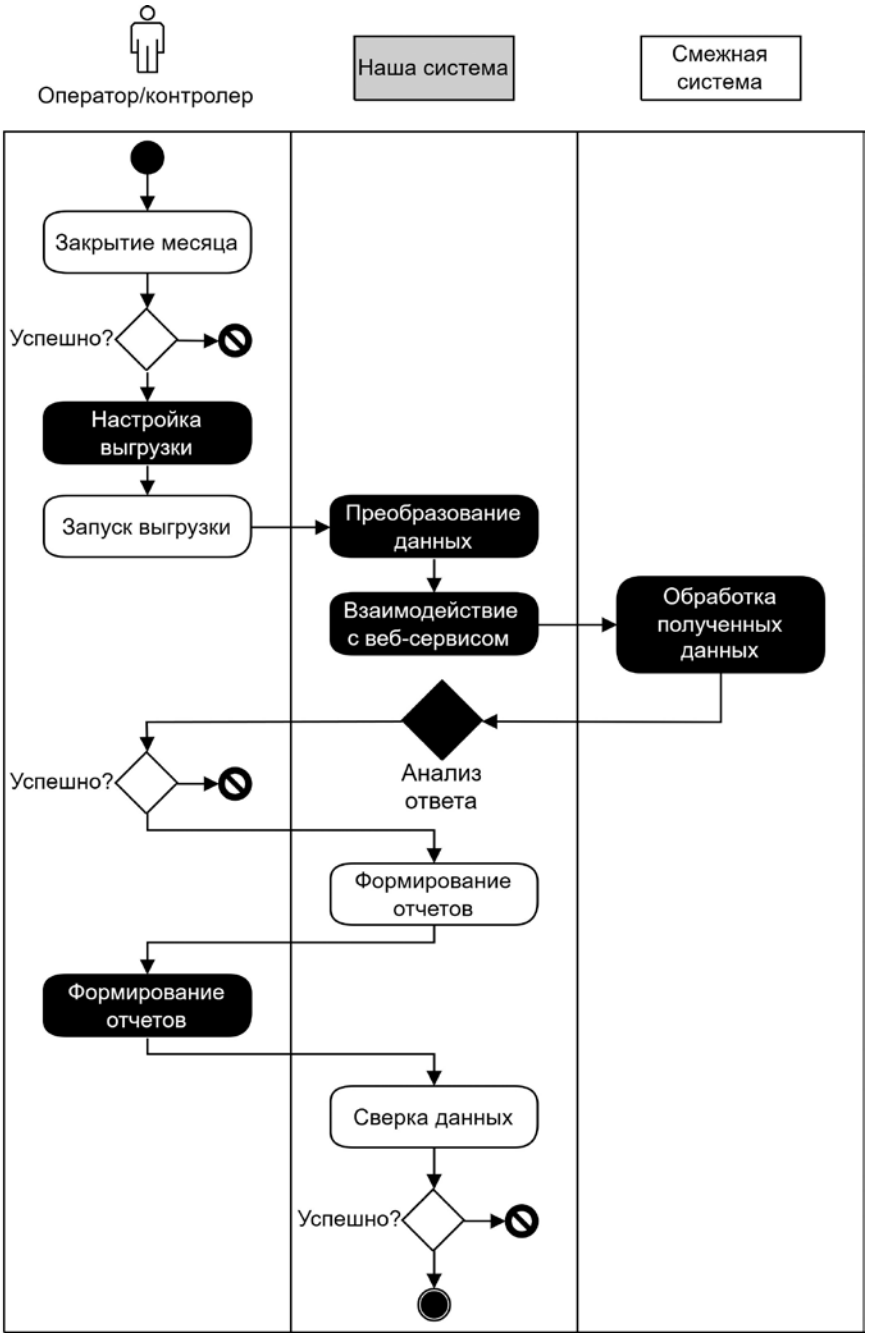

Рис. 3.2. Диаграмма хронометрического черного ящика

Листинг 4.1. Примитивные операции с примитивными типами

```
// При сложении платформа ориентируется на тип самой первой величины.

А = "Весна";
Б = 2023;
Результат = А + Б;
// Результат: "Весна2023".

Результат = Б + А;
// Получим ошибку преобразования типов, строку нельзя прибавить к числу.

Г = '20230101';
Результат = Г + Б;
// Результат: 01.01.2023 0:33:43.

Результат = Б + Г;
// Получим ошибку, к числу можно прибавить только число.

// Если правый операнд равенства можно интерпретировать как логическое
// условие, платформа так и делает.

Результат = А = Б;
// Результат: Ложь.

Результат = Б > Г;
// Получим ошибку, на больше/меньше можно сравнивать только значения
// одного типа.

В = "Веялка";
Результат = А > В;
// Результат: Истина, строки сравниваются "в порядке сортировки".

// Ну и так далее.
```

Листинг 4.2. Неявное преобразование типов платформой

```
Док = Документы.НашаОперация.СоздатьДокумент();
Док.Дата = "20230323";
// Док.Дата: 23.03.2023 0:00:00.
// Но со строкой "23.03.2023" так не получится, будет пустая дата.
Док.Сумма = "7.62";
// Сумма -> Число(15,2).
// Док.Сумма: 7,62, с числами такой трюк тоже работает.
```

Листинг 4.3. Явное преобразование к простому типу

```
Результат = Строка(7.62);
// Результат: "7,62".
Результат = Дата("20230323");
// Результат: 23.03.2023  0:00:00.
Результат = Дата("01.03.2003");
// Получим ошибку, с датой так не работает.
Результат = Число("-00000.00000");
// Результат: 0, это было вполне годное число.
Результат = Число("Ноль");
// Получим ошибку, с числами так тоже не работает.
```

Листинг 4.4. Варианты преобразования к типу Булево

```
Коллекция = Новый Структура("Раз,Два,Три");
Если Коллекция Тогда
    // Получим ошибку, что угодно в булево не преобразовать.
КонецЕсли;
К = Коллекция.Количество();
Если К И К - 5 Тогда
    Сообщить("Любое ненулевое число считается Истиной");
    // Что довольно странно.
КонецЕсли;
```

Листинг 4.5. Брутальное преобразование к типу Число

```
// Пожалуйста, никогда так не делайте.
Попытка
    Результат = Число(ВхДанные);
Исключение
    Результат = 0;
КонецПопытки;
```

Листинг 4.6. Переусложненное преобразование к типу Число

```
// Выполняет безопасное умное преобразование к типу Число.
//
// Параметры:
//    ВхДанные - Произвольный - Исходная величина (любая).
//
// Возвращаемое значение:
//    Число - Результат преобразования.
//
Функция ЭкстраЧисло(ВхДанные) Экспорт
    // Сперва преобразуем к строке, это безопасно.
    Заготовка = Строка(ВхДанные);
    Если ПустаяСтрока(Заготовка) Тогда
        Возврат 0; // Там ничего и не было.
    КонецЕсли;
    // Последовательно выполняем промежуточные преобразования и проверки:
    //  - выбрасываем все недопустимые в числе символы;
    //  - проверяем, минус может стоять только на первой позиции;
    //  - проверяем, точка может быть только одна;
    //  - проверяем, точкой может работать запятая;
    //  - убираем ведущие нули в целой части;
    //  - убираем нули в конце дробной части;
    //  - проверяем, после точки может быть только N разрядов;
    //  - ну и так далее, пока не надоест.
    Результат = Число(Заготовка);
    Возврат Результат;
КонецФункции // ЭкстраЧисло()
```

Листинг 4.7. Служебный метод удаления «нечисловых» символов

```
// Удаляет из переданной строки все "нечисловые" символы.
// На порядок следования этих символов внимания не обращает.
//
// Параметры:
//    ВхДанные – Произвольный – Которые нужно обработать.
//    СтрогийРежим – Булево – Любой "мусорный" символ -> вернуть "".
//
// Возвращаемое значение:
//    Строка – Результат обработки.
//
Функция ЗаготовкаЧисла(ВхДанные, СтрогийРежим = Ложь) Экспорт
    Заготовка = Строка(ВхДанные);
    Если ПустаяСтрока(Заготовка) Тогда
        Возврат "";
    КонецЕсли;
    БуквыАлфавита = "-.0123456789";
    Алфавит = Новый Соответствие;
    Для Счетчик = 1 По СтрДлина(БуквыАлфавита) Цикл
        Буква = Сред(БуквыАлфавита,Счетчик,1);
        Алфавит.Вставить(Буква,Истина);
    КонецЦикла;
    Результат = "";
    Для Счетчик = 1 По СтрДлина(Заготовка) Цикл
        Буква = Сред(Заготовка,Счетчик,1);
        ЭтоЧисловойСимвол = (Алфавит.Получить(Буква) = Истина);
        Если ЭтоЧисловойСимвол Тогда
            Результат = Результат + Буква;
        ИначеЕсли СтрогийРежим Тогда
            Возврат "";
        КонецЕсли;
    КонецЦикла;
    Возврат Результат;
КонецФункции // ЗаготовкаЧисла()
```

Листинг 4.8. Безопасное преобразование к типу Число

```
// Выполняет относительно безопасное преобразование переданного значения
// в тип Число. Наличие любого "мусорного" символа интерпретирует как
// признак нуля.
//
// Параметры:
//    ВхДанные – Произвольный – Которые нужно преобразовать.
//
// Возвращаемое значение:
//    Число – Результат преобразования.
//
Функция КакЧисло(ВхДанные) Экспорт
    Заготовка = ЗаготовкаЧисла(ВхДанные, Истина);
    Если ПустаяСтрока(Заготовка) Тогда
        Возврат 0;
    КонецЕсли;
    Результат = 0;
    Попытка
        Результат = Число(Заготовка);
    Исключение
        // Просим прощения у коллег по цеху за лишний except.
    КонецПопытки;
    Возврат Результат;
КонецФункции // КакЧисло()
```

Листинг 4.9. Хитрое преобразование к типу Число на стороне сервера

```
// Преобразуем что угодно, любое даже не к числу, а сразу к сумме.
Буфер = Новый ТаблицаЗначений;
КвЧ = Новый КвалификаторыЧисла(15, 2, ДопустимыйЗнак.Любой);
Буфер.Колонки.Добавить("Сумма",Новый ОписаниеТипов("Число",КвЧ));
Элемент = Буфер.Добавить();
Элемент.Сумма = ВхДанные;
Результат = Элемент.Сумма;
```

Листинг 4.10. Чтение реквизита ссылочной переменной «через точку»

```
// Некая обработка данных, прикладной контекст неважен.
//
// Параметры:
//    Накладная – ДокументСсылка.РасходнаяНакладная – Предмет обработки.
//
Процедура СделатьЧтоТоХорошее(Накладная) Экспорт
    НашКлиент = Накладная.Контрагент;
    НашаДата = Накладная.ДатаОтгрузки;
    // Дальше выполняется какой-то алгоритм.
КонецПроцедуры // СделатьЧтоТоХорошее
```

Листинг 4.11. Чтение «через точку» с предварительным извлечением объекта

```
НашДокумент = Накладная.ПолучитьОбъект();
НашКлиент = НашДокумент.Контрагент;
НашаДата = НашДокумент.ДатаОтгрузки;
// Дальше выполняется какой-то алгоритм.
```

Листинг 4.12. Чтение реквизита ссылочной переменной запросом

```
Запрос = Новый Запрос;
Запрос.Текст =
"ВЫБРАТЬ
|    РасходнаяНакладная.Контрагент КАК Контрагент,
|    РасходнаяНакладная.ДатаОтгрузки КАК ДатаОтгрузки
|ИЗ
|    Документ.РасходнаяНакладная КАК РасходнаяНакладная
|ГДЕ
|    РасходнаяНакладная.Ссылка = &Ссылка";
Запрос.УстановитьПараметр("Ссылка", Накладная);
Выборка = Запрос.Выполнить().Выбрать();
Если Выборка.Следующий() Тогда
    НашКлиент = Выборка.Контрагент;
    НашаДата = Выборка.ДатаОтгрузки;
Иначе
    Комментарий = "Не удалось найти документ " + Накладная;
    ВызватьИсключение Комментарий; // Внутренняя ошибка.
КонецЕсли;
```

Листинг 4.13. Чтение реквизита ссылочной переменной библиотечным методом

```
НашКлиент = ОбщегоНазначения.ЗначениеРеквизитаОбъекта(Накладная,
"Контрагент");
НашаДата = ОбщегоНазначения.ЗначениеРеквизитаОбъекта(Накладная,
"ДатаОтгрузки");
```

Листинг 4.14. Формат хранения ссылок в базе данных

```
// Это, конечно, не дословное представление, а упрощенное, схематичное.
>SELECTIDFROMDocument42

ID
----------
00235ebd-2b50-4e90-be48-01bb3d43be64
f42b9d04-c0c3-4ecc-99f3-6853b3bc9a47
9c22f36e-3fb4-4ba0-b7e3-b3f73b2bada1
2ba4c043-3f5f-4184-bce3-6d72402e5c10
7e03cf77-62e5-40ea-9bfd-828cd72e3be1
f95e9ce6-c40f-4816-9fb9-461ff947044f
ed4405ce-6aa7-4e5b-9e23-c2200cc180d9
```

Листинг 4.15. Запись и чтение идентификатора объекта в качестве ссылки

```
Элемент = РегистрыСведений.Протоколы.СоздатьМенеджерЗаписи();
Элемент.ИдЗаписи = Строка(Новый УникальныйИдентификатор);
Элемент.ВидДокумента = "РасходнаяНакладная";
Элемент.Ссылка = Строка(Накладная.УникальныйИдентификатор());
Элемент.Записать(Истина);
```

Листинг 4.16. Получение идентификатора объекта через навигационную ссылку

```
// Записываем навигационную ссылку в объект данных.
Элемент = РегистрыСведений.Протоколы.СоздатьМенеджерЗаписи();
Элемент.ИдЗаписи = Строка(Новый УникальныйИдентификатор);
Элемент.ВидДокумента = "РасходнаяНакладная";
Элемент.НавСсылка = ПолучитьНавигационнуюСсылку(Накладная);
Элемент.Записать(Истина);
// Читаем навигационную ссылку из ранее записанного объекта.
Элемент = РегистрыСведений.Протоколы.СоздатьМенеджерЗаписи();
Элемент.ИдЗаписи = ИдЗаписи;
Элемент.Прочитать();

// Навигационная ссылка имеет вид:
// e1cib/data/Документ.
// РасходнаяНакладная?ref=966c204ef6e5f1e011edcc9a13871811
// Нужно преобразовать ID к правильному виду (для GUID).
// Лучше, конечно, вынести преобразование в отдельную сервисную функцию
НавИд = Прав(Элемент.НавСсылка,32);
Фраг1 = Сред(НавИд,25,8);
Фраг2 = Сред(НавИд,21,4);
Фраг3 = Сред(НавИд,17,4);
Фраг4 = Сред(НавИд,1,4);
Фраг5 = Сред(НавИд,5,12);
СтрИд = Фраг1 + "-" + Фраг2 + "-" + Фраг3 + "-" + Фраг4 + "-" + Фраг5;
Ид = Новый УникальныйИдентификатор(СтрИд);

// Находим ссылку на документ по его идентификатору.
Ссылка = Документы[Элемент.ВидДокумента].ПолучитьСсылку(Ид);
```

Листинг 4.17. Переход по навигационной ссылке к форме объекта

```
ПерейтиПоНавигационнойСсылке(НавСсылка);
// Естественно, это работает только на стороне клиента.
// И будет выполнен серверный вызов.
```

Листинг 4.18. Установка перекрестной ссылки между двумя объектами

```
Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)
    // Создадим элемент-расшифровку и установим перекрестные ссылки
    // между документом и элементом справочника.
    Если Не ЭтотОбъект.ЭтоНовый() Тогда
        Возврат;
    КонецЕсли;
    НашаСсылка = Документы.РасходнаяНакладная.ПолучитьСсылку();
    ЭтотОбъект.УстановитьСсылкуНового(НашаСсылка);
    Элемент = Справочники.РасшифровкиНакладных.СоздатьЭлемент();
    Элемент.Накладная = НашаСсылка;
    Элемент.Записать();
    ЭтотОбъект.Расшифровка = Элемент.Ссылка;
КонецПроцедуры // ПередЗаписью()
```

Листинг 4.19. Проверка на наличие конкретного ключа

```
// Структура позволяет моментально проверить наличие в ней
// конкретного ключа.
КлючНайден = Коллекция.Свойство(Ключ);

// А для соответствия это работает только тогда, когда по ключу в нем
// есть значение, иначе нужно перебирать всю коллекцию.
Значение = Коллекция.Получить(Ключ);
КлючНайден = Ложь;
Если Значение = Неопределено Тогда
    Для Каждого Элемент Из Коллекция Цикл
        Если Элемент.Ключ = Ключ Тогда
            КлючНайден = Истина;
            Прервать;
        КонецЕсли;
    КонецЦикла;
Иначе
    КлючНайден = Истина;
КонецЕсли;
```

Листинг 4.20. Заполнение параметров области табличного документа

```
// Готовим структуру, ключи которой совпадают
// с именами параметров области табличного документа.
Коллекция = Новый Структура;
Коллекция.Вставить("Контрагент", НашКлиент);
Коллекция.Вставить("ДатаОтгрузки", НашаДата);

// Получаем из макета область и заполняем  параметры.
Область = НашМакет.ПолучитьОбласть(ИмяМакета);
Область.Параметры.Заполнить(Коллекция);

// И выводим заполненную область в финальный документ.
Результат.Вывести(Область);
```

Листинг 4.21. Поиск строк в таблице значений

```
Отбор = Новый Структура;
Отбор.Вставить("Контрагент", НашКлиент);
Отбор.Вставить("ДатаОтгрузки", НашаДата);
НайденныйСтроки = Коллекция.НайтиСтроки(Отбор);
Если НайденныйСтроки.Количество() = 0 Тогда
    НовыйЭлемент = Коллекция.Добавить();
    ЗаполнитьЗначенияСвойств(НовыйЭлемент,Отбор);
    НовыйЭлемент.Накладная = НашаСсылка;
    // Ну и так далее по алгоритму.
КонецЕсли;
```

Листинг 4.22. Кэширование отработанных в цикле значений

```
ОтработанныеКлиенты = Новый Соответствие;
Пока Выборка.Следующий() Цикл
    НашКлиент = Выборка.Контрагент;
    Если ОтработанныеКлиенты.Получить(НашКлиент) = Истина Тогда
        Продолжить;
    КонецЕсли;
    // Выполняем какие-то операции с контрагентом и заносим его
    // в список отработанных.
    ОтработанныеКлиенты.Вставить(НашКлиент, Истина);
КонецЦикла;
```

Листинг 4.23. Ускорение поиска таблицы значений с помощью квазииндекса

```
// Предположим, что в нашей таблице значений (в числе прочих)
// есть две колонки:
//  - ЛицевойСчет, СправочникСсылка.ЛицевыеСчета.
//  - ДатаОплаты, ДатаВремя.
// Причем таблица устроена так, что комбинация ЛС/ДатаОплаты является
// уникальной. А строк в таблице может быть ну очень много.
// При заполнении таблицы формируем квазииндекс по нашим двум колонкам.
КвазиИндекс = Новый Соответствие;
Пока НекоеУсловиеЦикла Цикл
    // Здесь какой-то алгоритм обработки данных.
    Элемент = Коллекция.Добавить();
    Элемент.ЛицевойСчет = КакойТоЛицевойСчет;
    Элемент.ДатаОплаты = КакаяТоДатаОплаты;
    // И так далее все прочие колонки.
    ИдЛС = Строка(Элемент.ЛицевойСчет.УникальныйИдентификатор());
    Ключ = ИдЛС + "@" + Формат(Элемент.ДатаОплаты,"ДФ=yyyyMMdd-hhmmss");
    КвазиИндекс.Вставить(Ключ,Элемент);
КонецЦикла;
// Далее при необходимости по любой паре ЛС/Дата оплаты мы моментально
// находим (или не находим) соответствующую строку таблицы значений.
ИдЛС = Строка(НашЛицевойСчет.УникальныйИдентификатор());
Ключ = ИдЛС + "@" + Формат(НашаДатаОплаты,"ДФ=yyyyMMdd-hhmmss");
Элемент = КвазиИндекс.Получить(Ключ);
Если Элемент = Неопределено Тогда
    // Указанная комбинация ЛС и даты в таблице отсутствует.
КонецЕсли;
// Зачем городить весь этот огород — при очень большом количестве строк
// в ТЗ такой квазииндекс должен работать существенно быстрее, чем
// НайтиСтроки(), даже по индексированной таблице. Ну и еще раз —
// демонстрируется не оптимизация конкретной ТЗ, а общий принцип работы
// с кодом (по этому принципу, например, работает механика РАУЗ).
```

Листинг 4.24. Загрузка списка выбора для поля ввода на форме

```
Варианты = Новый Массив;
Варианты.Добавить(Ссылка1);
Варианты.Добавить(Ссылка2);
Варианты.Добавить(Ссылка3);
// Представим, что этот массив был получен из какого-то другого метода.
Элементы.ВариантАлгоритма.СписокВыбора.ЗагрузитьЗначения(Варианты);
```

Листинг 4.25. Проверка существования множества файлов

```
// Предположим, что у нас есть массив полных имен файлов
// и для каждого из них нам необходимо знать, существует ли такой файл.
// Всю информацию можно собрать в одном списке значений.
НашиФайлы = Новый СписокЗначений;
Для Каждого Имя Из ИменаФайлов Цикл
    Файл = Новый Файл(Имя);
    НашиФайлы.Добавить(Имя,,Файл.Существует());
КонецЦикла;
```

Листинг 4.26. Удаление элементов коллекции «по указателю»

```
// На первом проходе собираем массив элементов для удаления.
ЛишниеЭлементы = Новый Массив;
Для Каждого Элемент Из Коллекция Цикл
    Если ЭлементНужноУдалить Тогда
        ЛишниеЭлементы.Добавить(Элемент);
    КонецЕсли;
КонецЦикла;
// На втором проходе выполняем физическое удаление.
Для Каждого Элемент Из ЛишниеЭлементы Цикл
    Коллекция.Удалить(Элемент);
КонецЦикла;
```

Листинг 4.27. Удаление элементов коллекции «по индексу»

```
Счетчик = Элементы.Количество();
Пока Счетчик > 0 Цикл
    Если ЭлементНужноУдалить Тогда
        Коллекция.Удалить(Счетчик-1);
    КонецЕсли;
    Счетчик = Счетчик - 1;
КонецЦикла;
```

Листинг 4.28. Создание утечки памяти через циклическую ссылку

```
НашМассив = Новый Массив;
НашМассив.Добавить("Много благих пожеланий
                   |и переусложненного программного кода");
// Создаем циклическую ссылку:
НашМассив.Добавить(НашМассив);
// Еще больше благих пожеланий и переусложненного программного кода.
```

Листинг 4.29. Своевременный разрыв циклической ссылки

```
НашМассив = Новый Массив;
НашМассив.Добавить("Много благих пожеланий
                   |и переусложненного программного кода");
// Создаем циклическую ссылку:
НашМассив.Добавить(НашМассив);
// Еще больше благих пожеланий и переусложненного программного кода.
// Разрываем циклическую ссылку:
НашМассив.Очистить();
// Угроза утечки памяти ликвидирована, можно жить дальше.
НашМассив.Добавить("Начинаем переусложнять код с новыми силами");
```

Листинг 4.30. Простейшая настройка технологического журнала

```
<?xmlversion="1.0"?>
<configxmlns="http://v8.1c.ru/v8/tech-log">
<loglocation="/var/log/1c/logs/excp" history="24">
<event>
<eqproperty="name" value="excp"/>
</event>
<propertyname="all"/>
</log>
<loglocation="/var/log/1c/logs/vrs" history="24">
<event>
<eqproperty="name" value="vrsrequest"/>
</event>
<event>
<eqproperty="name" value="vrsresponse"/>
</event>
<propertyname="all"/>
</log>
<dumplocation="/var/log/1c/dumps" create="1" type="3"/>
</config>
```

Листинг 4.31. Примитивное сравнение даты с текущей

```
Для Каждого Ссылка Из КоллекцияНакладных Цикл
    ДатаОтгрузки = ОбщегоНазначения.ЗначениеРеквизитаОбъекта(Ссылка,
    "ДатаОтгрузки");
    Если ДатаОтгрузки > ТекущаяДата() Тогда
        ВыполнитьНеобходимыеДействия();
    КонецЕсли;
КонецЦикла;
```

Листинг 4.32. Использование текущей даты сеанса

```
СегодняшняяДата = ТекущаяДатаСеанса();
Для Каждого Ссылка Из КоллекцияНакладных Цикл
    ДатаОтгрузки = ОбщегоНазначения.ЗначениеРеквизитаОбъекта(Ссылка,
    "ДатаОтгрузки");
    Если НачалоДня(ДатаОтгрузки) > НачалоДня(СегодняшняяДата) Тогда
        ВыполнитьНеобходимыеДействия();
    КонецЕсли;
КонецЦикла;
```

Листинг 4.33. Работа с часовым поясом сеанса/информационной базы

```
// Часовой пояс сеанса/ИБ в любой момент можно установить:
ИмяПояса = "Asia/Krasnoyarsk";
УстановитьЧасовойПоясИнформационнойБазы(ИмяПояса);
ИмяПояса = "Europe/Kaliningrad";
УстановитьЧасовойПоясСеанса(ИмяПояса);
// Можно прочитать ранее установленные значения:
ПоясИБ = ПолучитьЧасовойПоясИнформационнойБазы();
ПоясСеанса = ЧасовойПоясСеанса();
// При желании можно узнать, в каком часовом поясе живет
// информационная система:
ПоясОС = ЧасовойПояс();
// (А вот установить этот параметр по понятным причинам не получится).
```

Листинг 4.34. Работа с текущими универсальными датой/временем

```
// Локальная отметка времени зависит от региональных установок ОС.
ЛокальнаяДата = ТекущаяДата();
// "Серверная" отметка времени зависит от установок сервера, ИБ и сеанса
СерверныйДата = ТекущаяДатаСеанса();
// Универсальные отметки времени будут абсолютно одинаково прочитаны
// вне зависимости от региональных настроек клиента, сеанса,
// ИБ и сервера.
ОбщаяДата = ТекущаяУниверсальнаяДата();
Таймштамп = ТекущаяУниверсальнаяДатаВМиллисекундах();
// Общую универсальную дату в миллисекундах (Число) бывает удобно
// использовать для тонкой сортировки очень коротких событий (несколько
// событий умещается в одну секунду, например, в журналах и протоколах).
```

Листинг 4.35. Преобразование даты/времени между часовыми поясами

```
// Это текущее время на компьютере, где живет рабочий процесс.
НашаПозиция = ТекущаяДата();
// А это часовой пояс компьютера, где живет рабочий процесс.
Москва = ЧасовойПояс();
// А это список всех возможных часовых поясов планеты Земля.
ВсеПояса = ПолучитьДопустимыеЧасовыеПояса();
// А вот так называется часовой пояс Калининграда.
Калининград = "Europe/Kaliningrad";
// Но у него есть и человеческое представление.
КалининградСтрокой = ПредставлениеЧасовогоПояса(Калининград);
// Часовой пояс Калининграда — это GMT+2.
СмещениеКалининграда = СмещениеСтандартногоВремени(Калининград);
// Чтобы не запутаться, существует общепланетарная позиция во времени.
ТочнаяПозиция = ТекущаяУниверсальнаяДата();
// Местное время в любой точке планеты можно получить простым
// сложением координат.
ВремяКалининграда = ТочнаяПозиция + СмещениеКалининграда;
НашеСмещение = СмещениеСтандартногоВремени(Москва);
НашеВремя = ТочнаяПозиция + НашеСмещение;
// А можно поручить эти вычисления любимой технологической платформе.
ВремяКалининграда = МестноеВремя(ТочнаяПозиция, Калининград);
ТочнаяПозиция = УниверсальноеВремя(НашеВремя, Москва);
// При желании можно уточнить универсальное время до третьего знака.
СовсемТочнаяПозиция = ТекущаяУниверсальнаяДатаВМиллисекундах();
// Ну и так далее.
```

Листинг 4.36. Недопустимый пример парных констант в конфигурации

```
ИмяНашейКонфигурации
  |--Константы
    |--ВключитьРежимУчетаПоПартиям - Булево
    |--НеИспользоватьПартионныйУчет – Булево
```

Листинг 4.37. Включение регламентного задания при записи константы

```
Процедура ПриЗаписи(Отказ)
    ПодсистемаВключена = ЭтотОбъект.Значение;
    Если ПодсистемаВключена Тогда
        ТрекингЗадачСервер.ВключитьРассылкуУведомленийИсполнителя();
    Иначе
        ТрекингЗадачСервер.ОтключитьРассылкуУведомленийИсполнителя();
    КонецЕсли;
КонецПроцедуры // ПриЗаписи()
```

Листинг 4.38. Метод-перехватчик для чтения константы

```
// Возвращает установленный для веб-сервиса системы XXX тайм-аут
// (значение по умолчанию - минута).
//
// Параметры:
//    Нет.
//
// Возвращаемое значение:
//    Число – Тайм-аут ожидания веб-сервиса (в секундах).
//
Функция ТаймаутВебСервисаРегистрацииПоручений  () Экспорт
    Таймаут = Константы.ТаймаутВебСервисаРегистрацииПоручений.Получить();
    Если Таймаут = 0 Тогда
        Таймаут = 60; // Значение по умолчанию.
    КонецЕсли;
    Возврат Таймаут;
КонецФункции // ТаймаутВебСервисаРегистрацииПоручений()
```

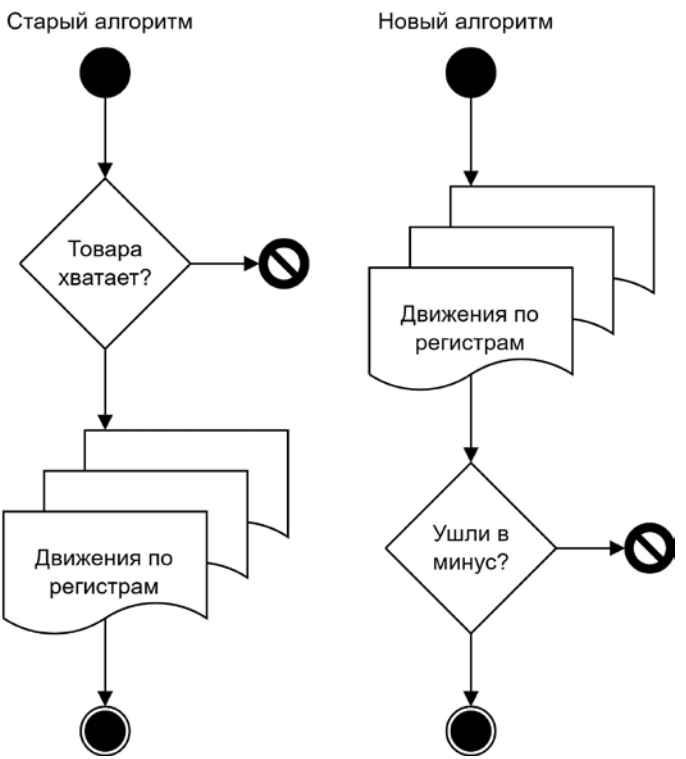

Рис. 4.1. Упрощенная блок-схема алгоритма проверки остатков

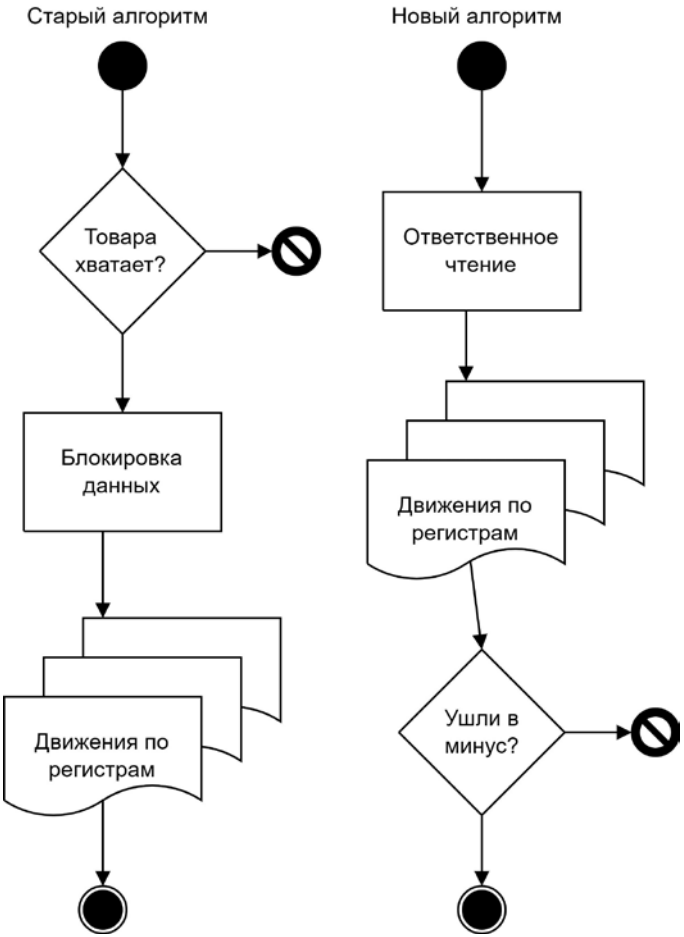

Рис. 4.2. Блок-схема алгоритма проверки остатков с учетом блокировок

Листинг 4.39. Установка свойства БлокироватьДляИзменения

```
Движения.ОстаткиПоКредитнымЛимитам.БлокироватьДляИзменения = Истина;
// Теперь можно выполнять движения по регистру, не опасаясь
// шальной взаимоблокировки.
```

Листинг 4.40. Универсальная запись элемента настройки в регистр сведений

```
// Добавляет в регистр новую запись для указанного
// ключа/значения настройки.
//
// Параметры:
//    Ключ – Строка – Уникальный ключ настройки.
//    Значение – Характеристика.КлючиНастроекПодсистемы – Новое значение
//                                                        настройки.
//
// Возвращаемое значение:
//    Булево – Результат операции (Истина = Настройка успешно записана).
//
Функция НовоеЗначениеНастройки(Ключ, Значение) Экспорт
    Ссылка = ПланыВидовХарактеристик.КлючиНастроекПодсистемы.КлючПоИмени
   (Ключ);
    Если Ссылка = Неопределено Тогда
        Возврат Ложь; // Такой настройки нет в ПВХ.
    КонецЕсли;
    ЭтоБезопасноеХранение =
    ПланыВидовХарактеристик.КлючиНастроекПодсистемы.ЭтоБезопасноеХранение
   (Ссылка);
    Если ЭтоБезопасноеХранение Тогда
        ОбщегоНазначения.ЗаписатьДанныеВБезопасноеХранилище(Ссылка,
        Значение, Ссылка.Код);
        Возврат Истина;
    КонецЕсли;
    Менеджер = РегистрыСведений.НастройкиПодсистемы.СоздатьМенеджерЗаписи()
    Менеджер.Период = ТекущаяДатаСеанса();
    Менеджер.КлючНастройки = Ссылка;
    Менеджер.ЗначениеНастройки = Значение;
    Менеджер.Автор = Пользователи.ТекущийПользователь();
    Менеджер.Записать(Истина);
    Возврат Истина;
КонецФункции // НовоеЗначениеНастройки()
```

Листинг 4.41. Чтение одиночного элемента настройки из регистра

```
// Возвращает значение настройки по указанной ссылке на элемент ПВХ.
//
// Параметры:
//    Ссылка – ПланВидовХарактеристикСсылка.КлючиНастроекПодсистемы –
//    Ссылочный ключ настройки, которую нужно найти.
//
// Возвращаемое значение:
//    Характеристика.КлючиНастроекПодсистемы – Найденное значение
//                                             настройки.
//    Не определено – Настройку найти не удалось.
//
Функция ЗначениеПоСсылке(Ссылка) Экспорт
    ЭтоБезопасноеХранение =
  ПланыВидовХарактеристик.КлючиНастроекПодсистемы.ЭтоБезопасноеХранение
  (Ссылка);
    Если ЭтоБезопасноеХранение Тогда
        Результат =
        ОбщегоНазначения.ПрочитатьДанныеИзБезопасногоХранилища(Ссылка,
        Ссылка.Код);
        Возврат Результат;
    КонецЕсли;
    Запрос = Новый Запрос;
    Запрос.Текст =
    "ВЫБРАТЬ
    |    Т.ЗначениеНастройки КАК ЗначениеНастройки
    |ИЗ
    |    РегистрСведений.НастройкиПодсистемы.СрезПоследних КАК Т
    |ГДЕ
    |    Т.КлючНастройки = &КлючНастройки";
    Запрос.УстановитьПараметр("КлючНастройки", Ссылка);
    Выборка = Запрос.Выполнить().Выбрать();
    Если Выборка.Следующий() Тогда
        Возврат Выборка.ЗначениеНастройки;
    КонецЕсли;
    Возврат Неопределено;
КонецФункции // ЗначениеПоСсылке()
```

Листинг 4.42. Чтение флага разрешения на запуск регламентных заданий

```
// Проверяет наличие в регистре сведений записи,
// указывающей на запрет запуска регламентного задания.
//
// Параметры:
//    ИмяЗадания - Строка - Которое необходимо проверить.
//
// Возвращаемое значение:
//    Булево — Результат проверки, Истина = Запрета нет.
//
Функция ЗапускРегламентаРазрешен(ИмяЗадания) Экспорт
    Запрос = Новый Запрос;
    Запрос.Текст =
    "ВЫБРАТЬ
    |    Т.ИмяЗадания КАК ИмяЗадания
    |ИЗ
    |    РегистрСведений.ФлагиЗапретаРегламентныхЗаданий КАК Т
    |ГДЕ
    |    Т.ИмяЗадания = &ИмяЗадания";
    Запрос.УстановитьПараметр("ИмяЗадания",ИмяЗадания);
    Результат = Запрос.Выполнить();
    Возврат Результат.Пустой();
КонецФункции // ЗапускРегламентаРазрешен()
```

Листинг 4.43. Проверка допустимости запуска самим регламентным заданием

```
Процедура РассылкаУведомленийИсполнителямЗадач() Экспорт
    Имя =
Метаданные.РегламентныеЗадания.РассылкаУведомленийИсполнителямЗадач.Имя;
    ЗапускРазрешен = ЗапускРегламентаРазрешен(Имя);
    Если Не ЗапускРазрешен Тогда
        Возврат;
    КонецЕсли;
    // Далее — штатный алгоритм регламентного задания.
КонецПроцедуры // РассылкаУведомленийИсполнителямЗадач()
```

Листинг 4.44. Создание регламентного задания с ключом

```
Задание =
Метаданные.РегламентныеЗадания.РассылкаУведомленийИсполнителямЗадач;
ЭкземплярЗадания = РегламентныеЗадания.СоздатьРегламентноеЗадание(Задание);
// Мы хотим, чтобы этот экземпляр регламентного задания не перекрывался с
// другими в пределах некоего множества, определяемого уникальным ключом.
ЭкземплярЗадания.Ключ = НашУникальныйКлюч; // Был задан где-то выше.
ЭкземплярЗадания.Записать();
```

Листинг 4.45. Поиск фонового задания по ключу

```
// Выполняет поиск фонового задания по указанной паре РЗ/Ключ.
//
// Параметры:
//    ИмяЗадания – Строка – Регламентное задание.
//    Ключ – Строка – Уникальный ключ экземпляра задания.
//
// Возвращаемое значение:
//    ФоновоеЗадание – Найденное задание.
//    Неопределено – Задание найти не удалось.
//
Функция ФоновоеЗаданиеПоКлючу(ИмяЗадания, Ключ) Экспорт
    Задание = Метаданные.РегламентныеЗадания.Найти(ИмяЗадания);
    Если Задание = Неопределено Тогда
        Возврат Неопределено; // Такого РЗ просто нет.
    КонецЕсли;
    Отбор = Новый Структура;
    Отбор.Вставить("РегламентноеЗадание", Задание);
    Отбор.Вставить("Ключ", Ключ);
    Отбор.Вставить("Состояние", СостояниеФоновогоЗадания.Активно);
    НайденныеФЗ = ФоновыеЗадания.ПолучитьФоновыеЗадания(Отбор);
    Если НайденныеФЗ.Количество() = 0 Тогда
        Возврат Неопределено;
    КонецЕсли;
    // Поскольку мы задействуем уникальный ключ,
    // такое активное задание должно быть только одно.
    Возврат НайденныеФЗ[0];
КонецФункции // ФоновоеЗаданиеПоКлючу()
```

Листинг 4.46. Проверка дублирования внутри самих фоновых заданий

```
Процедура РассылкаУведомленийИсполнителямЗадач() Экспорт
    Имя =
Метаданные.РегламентныеЗадания.РассылкаУведомленийИсполнителямЗадач.Имя;
    Ключ = НашУникальныйКлюч; // Был задан где-то выше.
    НайденноеФЗ = ФоновоеЗаданиеПоКлючу(Имя, Ключ);
    Если Не НайденноеФЗ = Неопределено Тогда
        // Фоновое задание с нашими ключами уже выполняется.
        // Обнаружен конфликт, и нужно что-то делать.
    КонецЕсли;
```

Листинг 4.47. Умная проверка дублирования внутри самих фоновых задан

```
Процедура РассылкаУведомленийИсполнителямЗадач() Экспорт
    Имя =
Метаданные.РегламентныеЗадания.РассылкаУведомленийИсполнителямЗадач.Имя;
    Ключ = НашУникальныйКлюч; // Был задан где-то выше.
    // Проверим, что код выполняется как фоновое задание.
    НашСеанс = ПолучитьТекущийСеансИнформационнойБазы();
    НашеФЗ = НашСеанс.ПолучитьФоновоеЗадание();
    Если Не НашеФЗ = Неопределено Тогда
        // Ищем другое фоновое задание с нашими ключами.
        НайденноеФЗ = ФоновоеЗаданиеПоКлючу(Имя, Ключ);
        Если Не НайденноеФЗ.УникальныйИдентификатор =
                НашеФЗ.УникальныйИдентификатор
        Тогда
            // Обнаружено уже запущенное фоновое задание, нужно что-то
            // с этим делать. Например, прервать выполнение нашего кода.
            // (Или выбросить исключение, если такая ситуация
            // указывает на ошибку.)
        КонецЕсли;
    КонецЕсли;
```

Листинг 4.48. Замена оператора ИЛИ на оператор В

```
// Исходный текст запроса:
|    РасходнаяНакладная.Статус = &Проверен
|    ИЛИ РасходнаяНакладная.Статус = &Подготовлен
// Оптимизированный текст запроса:
|    РасходнаяНакладная.Статус В(&ДопустимыеСтатусы)
```

ГЛАВА 5. НАХОДИМ ПРОЕКТНЫЕ РЕШЕНИЯ

Листинг 5.1. Директива НаКлиентеНаСервере (пример из типовой конфигурации)

```
// Честно говоря, не очень понятно, зачем делать бесконтекстный вызов
// с передачей контекста через параметр, но, видимо,
// какие-то причины для этого имелись.
&НаКлиентеНаСервереБезКонтекста
Функция РезультатРедактированияНачислений(Форма)
    Сотрудник = Форма.Сотрудник;
    ПорядокРасчета = Форма.ПорядокРасчета;
    СреднийЗаработок = Форма.СреднийЗаработок;
    РасшифровкаСреднегоЗаработка = Форма.РасшифровкаСреднегоЗаработка;
    МесяцыКорректировки = Форма.МесяцыКорректировки;
    // Далее некий алгоритм расчета.
```

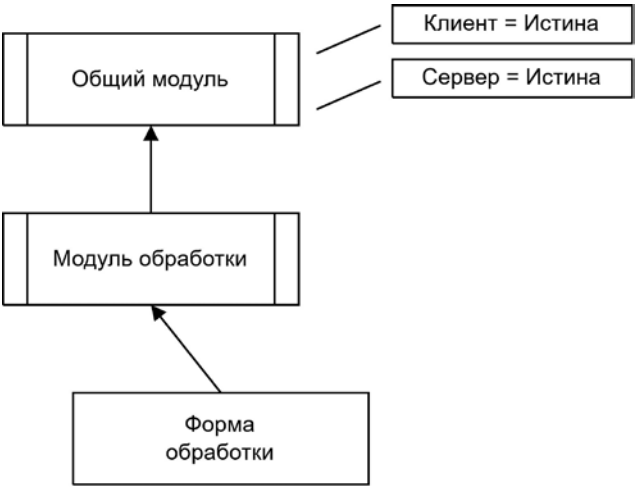

Рис. 5.1. Загадка клиент-серверного взаимодействия в режиме обычного приложения

Листинг 5.2. Типичная инструкция препроцессора по стандарту #680

```
// Код расположен в модуле объекта (это документ).
#Если Сервер Или ТолстыйКлиентОбычноеПриложение
             Или ВнешнееСоединение Тогда

Процедура ПередЗаписью(Отказ, РежимЗаписи, РежимПроведения)

// Ну и так далее, обработчики событий, экспортные методы, etc.

#Иначе
ВызватьИсключение НСтр("ru = 'Недопустимый вызов объекта на клиенте.'");
#КонецЕсли
```

Листинг 5.3. Вынос серверного кода из модуля формы в модуль менеджера

```
// В модуле формы обработки оставляем только вызов.
&НаСервере
Процедура ПриСозданииНаСервере(Отказ, СтандартнаяОбработка)
    Обработки.УправлениеИнтеграцией.ГлавнаяФорма_ПриСозданииНаСервере(
                                    ЭтотОбъект, Отказ, СтандартнаяОбработка);
КонецПроцедуры // ПриСозданииНаСервере()

// В модуле менеджера соответствующей обработки:
Процедура ГлавнаяФорма_ПриСозданииНаСервере(Форма,Отказ,
                                                СтандартнаяОбработка) Экспорт
    // Здесь располагаем собственно бизнес-логику формы.
КонецПроцедуры // ГлавнаяФорма_ПриСозданииНаСервере()
```

Листинг 5.4. Контекстный серверный вызов

```
&НаКлиенте
Процедура ОтменитьСессиюОбмена(Команда)
    ОтменитьСессиюОбменаНаСервере();
КонецПроцедуры // ОтменитьСессиюОбмена()

&НаСервере
Процедура ОтменитьСессиюОбменаНаСервере() Экспорт
    // Фактически мы протащили весь контекст вызова с клиента на сервер,
    // чтобы задействовать одну-единственную переменную.
    УправлениеИнтеграциейСервер.ОтменитьСессию(Объект.СессияОбменаДанными);
КонецПроцедуры // ОтменитьСессиюОбменаНаСервере()
```

Листинг 5.5. Бесконтекстный серверный вызов

```
&НаКлиенте
Процедура ОтменитьСессиюОбмена(Команда)
    ОтменитьСессиюОбменаНаСервере(Объект.СессияОбменаДанными);
КонецПроцедуры // ОтменитьСессиюОбмена()

&НаСервереБезКонтекста
Процедура ОтменитьСессиюОбменаНаСервере(Сессия)
    // Если нужна только одна переменная,
    // логичным будет только ее на сервер и пересылать.
    УправлениеИнтеграциейСервер.ОтменитьСессию(Сессия);
КонецПроцедуры // ОтменитьСессиюОбменаНаСервере()
```

Листинг 5.6. Пример очень простого потока событий

```
Пользователь запрашивает отчет по текущим задачам
    Через команду в разделе "Задачи"
    Альтернатива: Через гиперссылку на рабочем столе
                (если включен dashboard)
Система формирует список задач для исполнителя
    Выполняет компоновку данных по схеме
    Альтернатива: Читает из "операционного кэша" (если кэш включен)
    Ошибка: Пользователь не обладает правами "Исполнитель задач"
Система готовит сформированный отчет к показу
    Формирует табличный документ и помещает на форму
    Альтернатива: Сериализует данные отчета и записывает в
                "операционный кэш" (если кэш включен)
Пользователь анализирует список текущих задач
```

Листинг 5.7. Типичный пример асинхронного взаимодействия клиентских методов

```
&НаКлиенте
Процедура ПриОткрытии(Отказ)
    Если ЗаполнитьПриОткрытии Тогда
        ПодключитьОбработчикОжидания("ЗаполнитьДанныеФормыНаКлиенте",
                                     0.1, Истина);
        ЗаполнитьПриОткрытии = Ложь;
        Модифицированность = Истина;
    КонецЕсли;
КонецПроцедуры
```

Листинг 5.8. Модальный способ задать вопрос пользователю

```
&НаКлиенте
Процедура ВыполнитьОперацию()
    ТекстВопроса = "Действительно выполнить операцию?";
    Режим = РежимДиалогаВопрос.ДаНет;
    Ответ = Вопрос(ТекстВопроса, Режим);
    Если Ответ = КодВозвратаДиалога.Да Тогда
        ВыполнитьОперациюНаСервере();
    КонецЕсли;
КонецПроцедуры // ВыполнитьОперацию()
```

Листинг 5.9. Первая инкарнация асинхронных клиентских функций

```
&НаКлиенте
Процедура ВыполнитьОперацию()
    ТекстВопроса = "Действительно выполнить операцию?";
    Описание = Новый ОписаниеОповещения
("ПодтвердитьВыполнениеОперации",
                                        ЭтотОбъект);
    Режим = РежимДиалогаВопрос.ДаНет;
    ПоказатьВопрос(Описание, ТекстВопроса, Режим);
КонецПроцедуры // ВыполнитьОперацию()

&НаКлиенте
Процедура ПодтвердитьВыполнениеОперации(Ответ, ДопПараметры) Экспорт
    Если Ответ = КодВозвратаДиалога.Да Тогда
        ВыполнитьОперациюНаСервере();
    КонецЕсли;
КонецПроцедуры // ПодтвердитьВыполнениеОперации()
```

Листинг 5.10. Вторая инкарнация асинхронных клиентских функций

```
&НаКлиенте
Асинх Процедура ВыполнитьОперациюАсинх()
    ТекстВопроса = "Действительно выполнить операцию?";
    Режим = РежимДиалогаВопрос.ДаНет;
    ОбещаниеОтвета = ВопросАсинх(ТекстВопроса, Режим);
    Ответ = Ждать ОбещаниеОтвета;
    Если Ответ = КодВозвратаДиалога.Да Тогда
        ВыполнитьОперациюНаСервере();
    КонецЕсли;
КонецПроцедуры // ВыполнитьОперациюАсинх()
```

Листинг 5.11. Проверка корректности ИНН и СНИЛС перед записью объекта

```
&НаКлиенте
Процедура ПередЗаписью(Отказ, ПараметрыЗаписи)
    // Проверяем корректность введенного ИНН при создании нового элемента
    // справочника ФизическиеЛица (задействуем уже реализованный
    // в конфигурации ЗУП алгоритм проверки).
    ЗначениеРеквизита = Элементы["ФизлицоИНН"].ТекстРедактирования;
    ЗначениеПроверки = СокрЛП(ЗначениеРеквизита);
    РеквизитПустой = ПустаяСтрока(ЗначениеПроверки);
    Если РеквизитПустой Тогда
        Комментарий = "Не заполнено поле ИНН.";
        ОбщегоНазначенияКлиент.СообщитьПользователю(Комментарий,
        "Элементы.ФизлицоИНН","ФизическоеЛицо.ИНН");
        Отказ = Истина;
        Возврат;
    КонецЕсли;
    ФлагОтказаИНН = Не
РегламентированныеДанныеКлиентСервер.ИННСоответствуетТребованиям(
Элементы.ФизлицоИНН.ТекстРедактирования, Ложь, "");
    Если ФлагОтказаИНН Тогда
        Комментарий = "Указанное значение не является корректным ИНН.";
        ОбщегоНазначенияКлиент.СообщитьПользователю(
Комментарий,,"Элементы.ФизлицоИНН","ФизическоеЛицо.ИНН");
        Отказ = Истина;
        Возврат;
    КонецЕсли;
КонецПроцедуры // ПередЗаписью()
```

Листинг 5.12. Техническая аннотация к универсальному методу РезультатЗапроса

```
// Универсальная функция. Выполняет запрос с указанным текстом и
// параметрами, возвращает результат как величину указанного типа.
// Опционально может выполнить некоторые служебные/вспомогательные
// действия (описаны функцией РезультатЗапросаКонф()).
//
// Параметры:
//    ТекстЗапроса - Строка,ТекстовыйДокумент - Содержит текст запроса.
//    ПараметрыЗапроса - Произвольный,Структура - Содержит параметры
//                                                запроса. В случае
//                                                если параметр только
//                                                один, указывается сразу
//                                                его значение.
//    ВариантВозврата – Число – Тип возвращаемого значения,
//                              кодируется числом
//                              (см. ВариантВозвратаПоКоду()).
//    Параметры – Структура – Дополнительные параметры метода
//                            (опции выполнения запроса). Возможные ключи
//                            и значения по умолчанию
//                            см. РезультатЗапросаКонф().
//
// Возвращаемое значение:
//    РезультатЗапроса, ВыборкаИзРезультатаЗапроса, ТаблицаЗначений,
//    Структура, Массив, Булево, Произвольный – Результат выполнения
//    запроса в указанном виде (управляется параметром ВариантВозврата).
//
Функция РезультатЗапроса(ТекстЗапроса, ПараметрыЗапроса = Неопределено,
                         ВариантВозврата = Неопределено,
                         Параметры = Неопределено) Экспорт
```

Листинг 5.13. Варианты вызова универсального метода РезультатЗапроса

```
// Самый простой вариант использования метода:
ТекстЗапроса = "ВЫБРАТЬ Т.Ссылка КАК Ссылка ИЗ Справочник.Сотрудники
               |КАК Т ГДЕ
               |Т.ЭтоГруппа = ЛОЖЬ И Т.ПометкаУдаления = ЛОЖЬ";
ВсеСотрудники = глЗапрос.РезультатЗапроса(ТекстЗапроса);

// Вариант чуть посложнее — запрос с одним параметром:
ТекстЗапроса = "ВЫБРАТЬ Т.Ссылка КАК Ссылка ИЗ Справочник.ШтатныеЕдиницы
               |КАК Т ГДЕ
               |Т.Подразделение = &Подразделение";
ВсеСотрудники = глЗапрос.РезультатЗапроса(ТекстЗапроса, НашОтдел);

// Сложный случай — пакетный запрос с отдельной обработкой
// нескольких результатов:
ТекстЗапроса = ТекстЗапроса_СтруктураКомпании();
ПараметрыЗапроса = Новый Структура("ОтчетныйМесяц,ИсключитьВакансии",
НачалоМесяца(ТекущаяДатаСеанса()),Истина);
ПакетРезультатов = глЗапрос.РезультатЗапроса(ТекстЗапроса,
                                             ПараметрыЗапроса, 3);
ТекущийЛичныйСостав = ПакетРезультатов[4];

// Дополнить менеджер временных таблиц новыми таблицами:
Заготовка = Новый Структура("МенеджерВТ", НашМенеджерВТ);
Опции = глЗапрос.РезультатЗапросаКонф(Заготовка);
ТекстЗапроса = ТекстЗапроса_СтруктураКомпанииВТ();
ПараметрыЗапроса = Новый Структура("ОтчетныйМесяц,ИсключитьВакансии",
                                   НачалоМесяца(ТекущаяДатаСеанса()),Ложь);
ПакетРезультатов = глЗапрос.РезультатЗапроса(ТекстЗапроса, ПараметрыЗапроса,
                                             2, Опции);
// Ну и так далее.
```

Листинг 5.14. Элементарная реализация метода Version

```
// В модуле веб-сервиса прописываем Get- и Post-варианты метода,
// каждый из которых вызывает одну и ту же общую функцию.

Функция versionGET(Запрос)
    Ответ = ИнтеграцияСервер.ОтветВебСервиса_Версия(Запрос);
    Возврат Ответ;
КонецФункции
Функция versionPOST(Запрос)
    Ответ = ИнтеграцияСервер.ОтветВебСервиса_Версия(Запрос);
    Возврат Ответ;
КонецФункции

// В общем модуле прописываем элементарную функцию,
// которая просто возвращает текущий номер версии.

//ОбработчикметодаVersion(GET/POST)HTTP-сервисаTechAPI.
// Возвращает в любом случае 200 и текущий номер версии.
//
// Параметры:
//    ВхЗапрос - HTTPСервисЗапрос - Входящий запрос HTTP-сервиса.
//
// Возвращаемое значение:
//    HTTPСервисОтвет - Сформированный ответ.
//
Функция ОтветВебСервиса_Версия(ВхЗапрос) Экспорт
    Ответ = Новый HTTPСервисОтвет(200);
    Ответ.УстановитьТелоИзСтроки(Метаданные.Версия);
    Возврат Ответ;
КонецФункции // ОтветВебСервиса_Версия()
```