# **FODB** - Flexible Object-oriented Data Base

Перейти:

# FODB - Flexible Object-oriented Data Base

*Продолжение…*

Перейти:



К главному меню…

## О концептуальных ограничениях

Ничего идеального – не бывает (это – аксиома).
Точно также и в нашей адаптивной и «СверхСуперКрутой» ИС есть, тем не менее, концептуальные ограничения системного характера.

Чтобы было понятно в чем мы ограничены, приведу некоторые сведения на стыке философии и ИИ.

Ранее эти сведения были весьма противоречивы и размазаны по множеству различных источников,
но нашелся человек (некто Варламов О.О., который, как я понимаю, является создателем компании «МИВАР» http://mivar.ru/ и http://science.mivar.ru/ ).
Который сумел это все обобщить и сделать следующий шаг еще в 2002-м году (написав книгу):

«ЭВОЛЮЦИОННЫЕ БАЗЫ ДАННЫХ И ЗНАНИЙ ДЛЯ АДАПТИВНОГО СИНТЕЗА ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ.
МИВАРНОЕ ИНФОРМАЦИОННОЕ ПРОСТРАНСТВО»

Бесценный, в нашей современности, труд (по моей личной оценке).
Там есть много спорных моментов, но это не умаляет ее ценности.

Мне эта книга попалась только, к сожалению, в августе-сентябре 2018-го. Т.е., поздновато.

И тем не менее, я почерпнул (и продолжаю это делать) из нее много чего для себя полезного
(у нас, кстати, есть эта книга).

К главному меню…

# О концептуальных ограничениях . **ВЕЩЬ и СВОЙСТВО**

Далее – просто некоторые цитаты из этой книги. Они вроде бы обо «всем давно известном и избитом еще с курса философии института», но они важны с точки зрения понимания того, что есть наша ИС.

«Три основных философских понятия, категории:
**вещь, свойство, отношение** - позволяют отобразить мир как бесконечное многообразие материальных реальностей в их возникновении, пребывании, изменении и гибели.»

*Примечание – мы используем вместо термина «вещь», термин «объект». И, кстати, лично я придерживаюсь точки зрения, что нет объектов, свойств и даже отношений, а есть непрерывные (и взаимодействующие) ПРОЦЕССЫ, распределенные как в пространстве, так и во времени.*
*Просто для нас (как «кратко-живущих» и весьма ограниченных) удобнее использовать «в обиходе» статистическое понятие «объект».*

«Таким образом, любая (всякая) определенная материальная реальность, существующая как нераздельность (цельность) многообразного - есть вещь. Тогда, вещью будет любой фрагмент материального мира.»

«Только **вещи** обладают самостоятельным бытием ... как нет и самих **свойств** и отношений вне и помимо вещей».

*Примечание – вот с этим утверждением (в контексте свойств) лично я категорически не согласен. Согласно моему пониманию (уж если на то пошло) любое свойство – есть объект.*
*Пример: цвет – это (вроде, как) свойство. Но это, также, объект, который имеет свои свойства (как минимум RGB, насыщенность).*

К главному меню…

# О концептуальных ограничениях . **МИВАР**

«МИВАР - Наименьший элемент трехмерного дискретного информационного пространства, которое образуется тремя осями: **<u>объектов</u>**, **<u>свойств</u>** и **<u>отношений</u>**.»

*Примечание1 – далее, идет введение в такое определение, как дискретное и неограниченное миварное пространство, которое является основой формирования ключевых вопросов, связанных с созданием адаптивных, эволюционных баз данных и знаний для синтеза интеллектуальных систем и, далее – путь к ИИ.*

*Примечание2 – далее, в этой книге, есть еще много, чего интересного, ценного и противоречивого (на мой взгляд),*
*но это уже для рассматриваемого вопроса – не так уж и важно.*

Всего этого (см. выше) достаточно, чтобы перейти к «нам». В данном случае – к МетаДанным.

К главному меню…

# О концептуальных ограничениях . **Метаданные**

**Метаданные** = «Множество типов объектов» + «Множество свойств» + «**Перечень видов свойств**»

Чуть позже мы еще детальнейшим (в пределах разумного) образом рассмотрим все, что связано с МетаДанными.
Поскольку «адаптивность» и «объектно-ориентированность» ИС базируется, в основном, на этом.

В контексте концептуальных ограничений.
Именно «Перечень видов свойств» определяет (в первую очередь) эти концептуальные ограничения.
А именно:
для отображения реалий предметной области (на уровне свойств) в информационные структуры ИС мы можем использовать только приведенные на рисунке виды свойств.

**Важно!**
Исключение составляет вид свойства «Неопределенно» (код = -1). Его использовать (в рамках текущей реализации) – не представляется возможным.

Виды свойств

| Код | Наименование | Мнемокод |
|---|---|---|
| -1 | Неопределенно | Unknown |
| 0 | Строка | String |
| 1 | Целое число | Integer |
| 2 | Вещественное число | Float |
| 3 | Дата | Date |
| 4 | Время | Time |
| 5 | Дата-Время | DateTime |
| 6 | Логическое | Boolean |
| 7 | Текст | Memo |
| 8 | Перечислимое | Enum |
| 9 | Ссылка на объект | LinkToObj |
| 10 | Ссылка на свойство | LinkToProp |
| 12 | Ссылка на справочник | LinkToDir |
| 103 | ПсевдоДата | PseudoDate |
| 104 | ПсевдоВремя | PseudoTime |
| 108 | ПсевдоПеречисление | PseudoEnum |

Ok Cancel

Почему они «появились» – понятно:
1. Этого было достаточно (по моему личному мнению) для решения качественных задач.
2. Физическое отсутствие времени на что-то «бОльшее».

К главному меню…

# Метаданные

**Метаданные** = «Множество типов объектов» + «Множество свойств» + «Перечень видов свойств»

**Множество типов объектов** - перечень всех разновидностей объектов, «известных» Системе.
Если появляются объекты нового типа, которые будут использоваться в дальнейшем, то новый тип должен быть сначала интерактивно внесен конечным пользователем в «множество типов» по определенным правилам.
Количество используемых типов объектов концептуально не ограничено.

*Применительно к миварному пространству рискну предположить, что типы объектов, это (в частности) одна из составляющих такого понятия, как «отношение между объектами».*

**Свойство** (объекта) – информационная сущность, определенным образом характеризующая информационный объект.
Свойство не может быть «охарактеризовано» другими свойствами и в этом его коренное отличие от объекта.
Свойства объектов могут быть:
- системно предопределенные (всегда присутствуют у любого объекта);
- определенные пользователем (задаются конечным пользователем).

**Множество свойств** – перечень всех свойств, «известных» Системе.
Если появляются новые (определенные пользователем) свойства, которые будут применяться в дальнейшем, то они должны быть сначала интерактивно внесены конечным пользователем в «множество свойств» по определенным правилам.
Количество определенных пользователем свойств концептуально не ограничено.

К главному меню…

# Метаданные

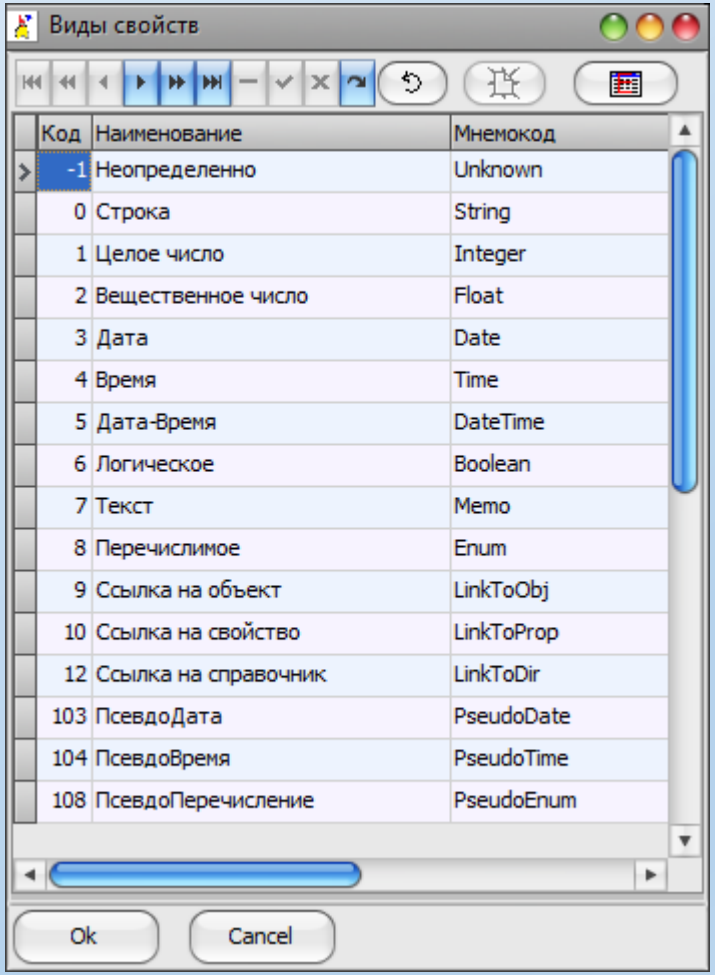
Когда конечный пользователь вносит новое свойство в Систему, то **вид свойства** обязательно должен быть выбран из следующего **перечня** (см. рисунок).

Исключение составляет вид свойства «Неопределенно» (код = -1).
Его использовать (в рамках текущей реализации) – не представляется возможным (установлен запрет на уровне ядра Системы).

*Примечание1 – вид свойства «Ссылка на свойство» (код = 10) сейчас не реализован (не используется).*
*Причина: на прикладном уровне это оказалось не востребованным,*
*а для «фундаментальности» - его существенно не достаточно.*

*Примечание2 – эти виды свойств (коды): 4, 5, 103, 104, 108 – реализованы слабо (не исчерпывающе корректно).*
*Причина та же: на прикладном уровне это ПОКА что оказалось не востребованным.*
*Но лично я уверен, что виды 103, 104 (и особенно 108) просто еще «не осознанны» конечными пользователями.*

*Применительно к миварному пространству рискну предположить,*
*что вид свойства - это, в частности, одна из составляющих такого понятия,*
*как «отношение между объектом и свойством».*




| Код | Наименование | Мнемокод |
|---|---|---|
| -1 | Неопределенно | Unknown |
| 0 | Строка | String |
| 1 | Целое число | Integer |
| 2 | Вещественное число | Float |
| 3 | Дата | Date |
| 4 | Время | Time |
| 5 | Дата-Время | DateTime |
| 6 | Логическое | Boolean |
| 7 | Текст | Memo |
| 8 | Перечислимое | Enum |
| 9 | Ссылка на объект | LinkToObj |
| 10 | Ссылка на свойство | LinkToProp |
| 12 | Ссылка на справочник | LinkToDir |
| 103 | ПсевдоДата | PseudoDate |
| 104 | ПсевдоВремя | PseudoTime |
| 108 | ПсевдоПеречисление | PseudoEnum |

К главному меню…

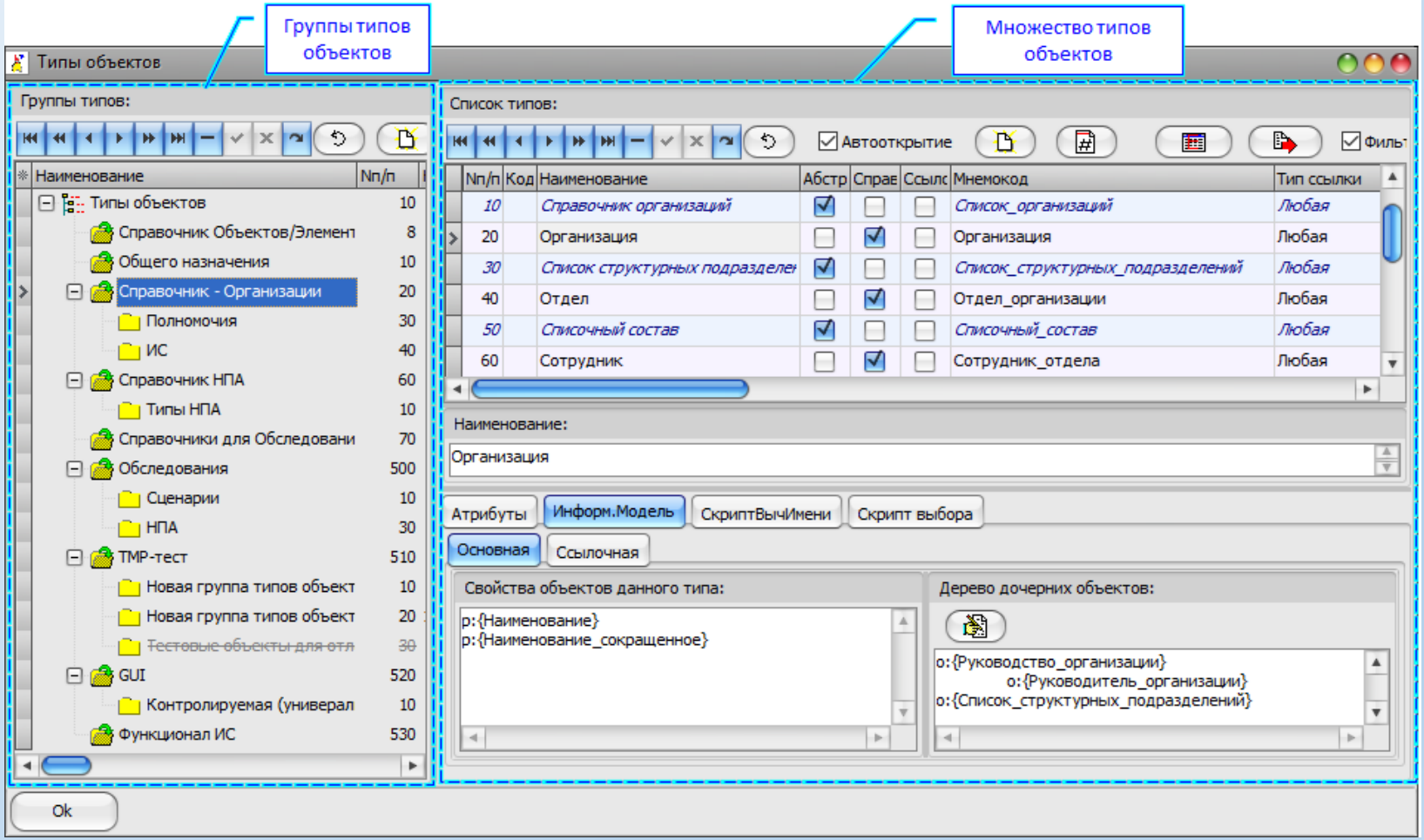
# Метаданные. **Множество типов объектов**

## Практическая реализация «множества типов объектов»

На уровне PostgreSQL-сервера:

1. Множество типов объектов (имя таблицы БД: «dir.ot_list»).
2. Группы типов объектов (имя таблицы БД: «dir.ot_grp»).
3. N-е количество ХФ.

На уровне GUI (см. рисунок):




К главному меню…

# Метаданные. **Множество типов объектов**

**Множество типов объектов** - перечень всех разновидностей объектов, «известных» Системе.
Если появляются объекты нового типа, которые будут использоваться в дальнейшем, то новый тип должен быть сначала интерактивно внесен конечным пользователем в «множество типов» по определенным правилам.
Количество используемых типов объектов концептуально не ограничено.

Основное назначение (в контексте объектной модели предметной области):

1. Строгая и однозначная классификация объектов по типовому признаку.
2. Хранение данных об информационной модели объектов данного типа.
3. Хранение скрипта, предназначенного для автоматического вычисления наименований объектов данного типа.
4. Системная поддержка одного из методов, реализующих механизм ссылочности объектов.
5. Хранение скрипта для выбора объектов данного типа (при использовании механизма ссылочности, реализованного на уровне ссылочных объектов) с применением механизма предметных указателей.

*Применительно к миварному пространству рискну предположить, что механизм ссылочности, это (в частности) одна из составляющих такого понятия, как «отношение между объектами».*

См., также, здесь…

**Важно!**

1. Нет обязательных требований, чтобы все типы объектов были «использованы» в процессе ввода информации о реальных объектах из предметной области.
2. Типы объектов используются, также, при настройке процессов миграции данных.

*Примечание – процесс миграции данных у нас реализован пока что только в части настроек и то, на уровне «декларации намерений».*
*Т.е., все осталось на уровне «мысленного эксперимента».*

К главному меню…

# Метаданные. **Множество типов объектов**

## Строгая и однозначная классификация объектов по типовому признаку.

Здесь все понятно, думаю.

Нормальная ИС (а тем более, объектно-ориентированная, да еще и наша) никогда не позволит превратить себя в мусорку.

**Важно!**

Каждый тип объекта имеет дополнительные атрибуты:

- признак абстрактности;
- признак местоположения в ИБ (объект или объект-справочник);
- тип объектов, на который могут ссылаться объекты данного типа при использовании механизма ссылочных объектов.

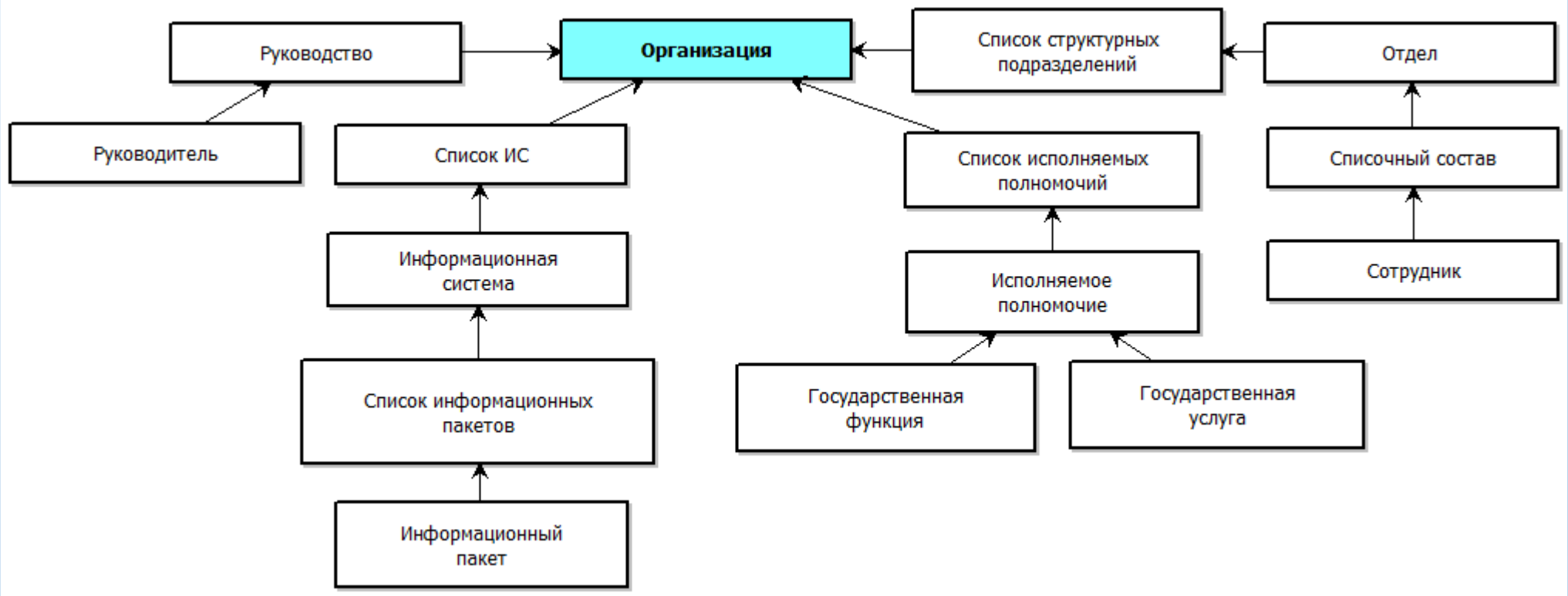
К главному меню…

Метаданные. **Множество типов объектов**

**Хранение данных об информационной модели объектов данного типа.**

Сначала – вводный пример.

Предположим, что Аналитик, в процессе изучения предметной области, выделил объекты определенного типа и выяснил их взаимосвязь. Получилось у него (на бумаге) вот так (см. рисунок).




К главному меню…

# Метаданные. **Множество типов объектов**

## Хранение данных об информационной модели объектов данного типа.

### Вводный пример.

При обычном (стандартном) подходе к разработке ИС, сразу же все понятно.

**1.** Аналитик (или разработчик) выделит (для создания) соответствующие таблицы БД (с соответствующей структурой полей).
Например, так (здесь могут быть варианты, но это неважно):
Организации; Руководители; Отделы; Сотрудники; Информационные системы; Информационные пакеты; Исполняемые полномочия; ГосУслуги/ГосФункции.

**2.** Учтут все реляционные связи (на уровне Master-Detail). Пропишут всю необходимую логику в триггерах.

**3.** «Нарисуют» все нужные и (что главное) КОНКРЕТНЫЕ GUI-формы.

**4.** Пропишут (на уровне исходных кодов) всю необходимую КОНКРЕТНУЮ логику для каждой КОНКРЕТНОЙ GUI-формы.

**5.** «Соберут» все это в виде приложения – и вперед, за наградами.

Но вот с адаптивной ИС такой подход не пройдет.
Поскольку заранее совершенно не ясно, какую именно предметную область будет обслуживать данная ИС.
Каким образом будут изменены информационные модели в процессе жизненного цикла ИС.
Т.е., заранее невозможно предсказать, что именно Аналитик нарисует на «бумаге».

К главному меню…

# Метаданные. **Множество типов объектов**

## Хранение данных об информационной модели объектов данного типа.

### Реализация в нашей ИС.

В нашей ИС Аналитик должен выполнить некоторые действия (по определенным правилам и с использованием конкретного инструментария), чтобы объяснить ИС - чего он от нее хочет.
И, кстати, ему совсем не обязательно рисовать все это на бумаге…
И выглядеть это будет так (см. рисунок).

```
o:{Руководство_организации}
 o:{Руководитель_организации}
o:{Список_структурных_подразделений}
 o:{Отдел_организации}
  o:{Списочный_состав}
   o:{Сотрудник_отдела}
o:{Список_ИС}
 o:{ИС}
  o:{Список_ИнформПакетов}
   o:{ИнформПакет}
o:{Список_исполняемых_полномочий}
 o:{Исполняемое_полномочие}
  o:{Сервис_полномочия_ГФ}
  o:{Сервис_полномочия_ГУ}
```

**Информационная модель** (шаблон информационной структуры) - это "описатель" (интерактивно сформированный пользователем по заданным правилам) информационной структуры, принятой по умолчанию для объектов заданного типа.

К главному меню…

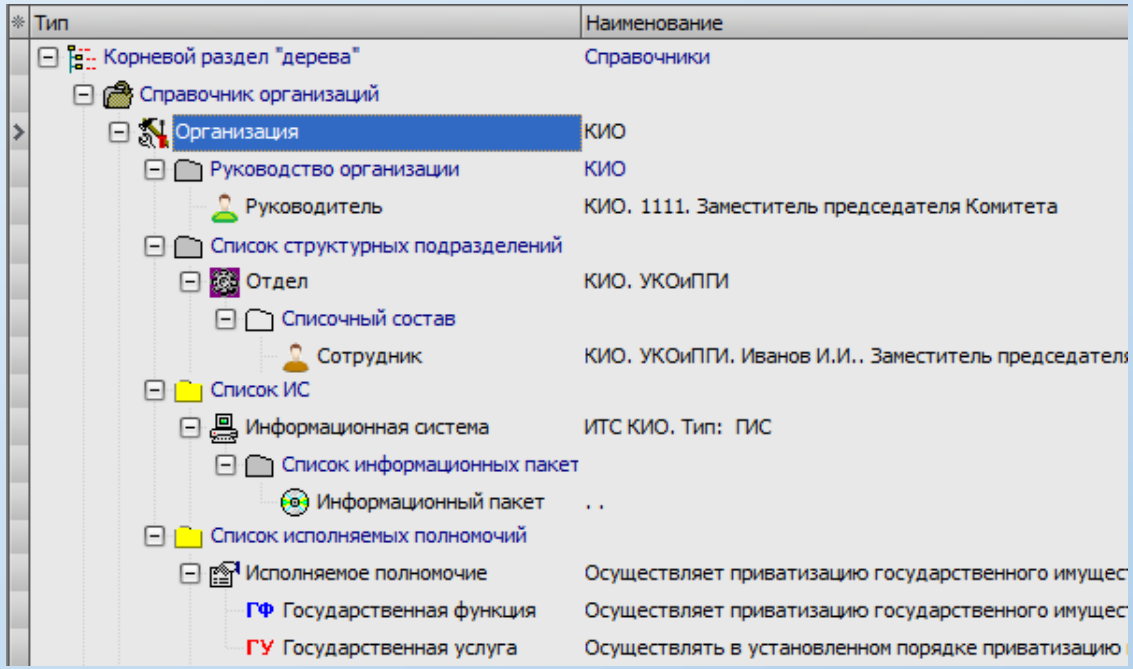
# Метаданные. **Множество типов объектов**

## Хранение данных об информационной модели объектов данного типа.

### Реализация в нашей ИС.

Теперь ИС совершенно точно знает: если конечный пользователь создаст объект типа «Организация», то «она» (ИС) должна сразу же (по умолчанию) создать иерархическую структуру в дереве объектов для нового объекта.

Т.е., при создании нового объекта типа «Организация» наша ИС создаст вот такую информационную структуру (см рисунок), но ясное дело, что пустую (в отличии от того, что на рисунке).




**Информационная структура** - это часть иерархического дерева объектов, начинающаяся с какого-либо родительского объекта, включая все дочерние объекты (и потомки), а также свойства этих объектов. Ее главное назначение - хранение индивидуальной (для данного объекта), четко классифицированной и структурированной информации приемлемого качества без концептуальных системных ограничений.

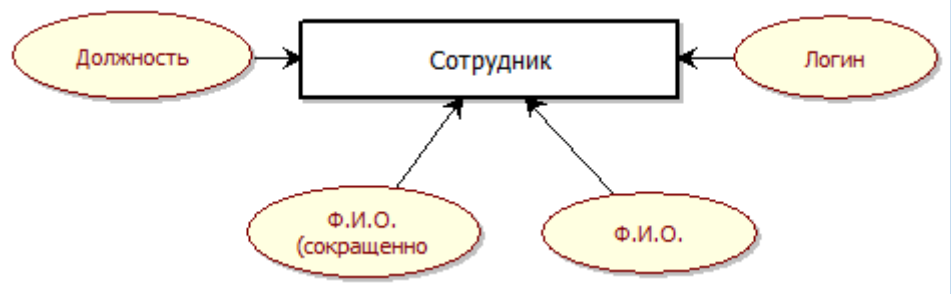
К главному меню…

# Метаданные. **Множество типов объектов**

## Хранение данных об информационной модели объектов данного типа.

### Реализация в нашей ИС.

Каждый объект может обладать свойствами.
Например, объект типа «Сотрудник» может иметь следующий набор свойств (см. рисунок).




В нашей ИС информационная модель содержит (факультативно), также, перечень свойств, владельцем которых являются объекты заданного типа. Перечень свойств (в информ. модели) также интерактивно формирует Аналитик по своему усмотрению. На рисунке ниже приведен иллюстрирующий пример.

```
p:{Должность}
p:{Ф_И_О}
p:{Ф_И_О_сокращенно}
p:{Login}
```

К главному меню…

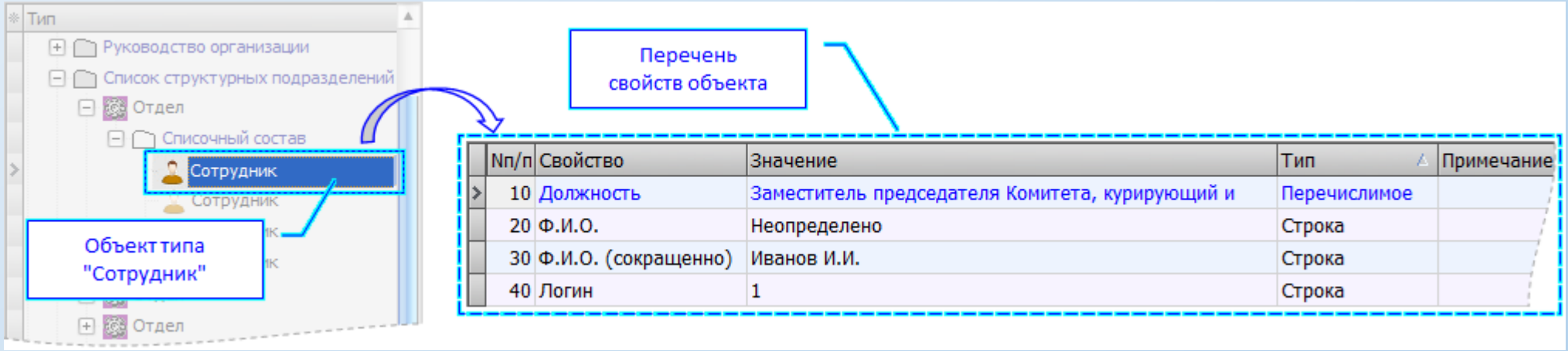
# Метаданные. **Множество типов объектов**

## Хранение данных об информационной модели объектов данного типа.

### Реализация в нашей ИС.

Теперь ИС совершенно точно знает: если конечный пользователь создаст объект типа «Сотрудник»,
то «она» (ИС) должна сразу же (по умолчанию) создать для нового объекта соответствующий перечень свойств.

Т.е., при создании <u>нового</u> объекта типа «Сотрудник» наша ИС создаст вот такой перечень свойств (см рисунок),
но ясное дело, что пустой (в отличии от того, что на рисунке).




| Nп/п | Свойство | Значение | Тип | Примечание |
|---|---|---|---|---|
| 10 | Должность | Заместитель председателя Комитета, курирующий и | Перечислимое | |
| 20 | Ф.И.О. | Неопределено | Строка | |
| 30 | Ф.И.О. (сокращенно) | Иванов И.И. | Строка | |
| 40 | Логин | 1 | Строка | |

Т.е., информационная модель – это некий шаблон, который использует ИС
для формирования соответствующих информационных структур для объектов соответствующего типа.
ВНЕ зависимости от конкретной предметной области.

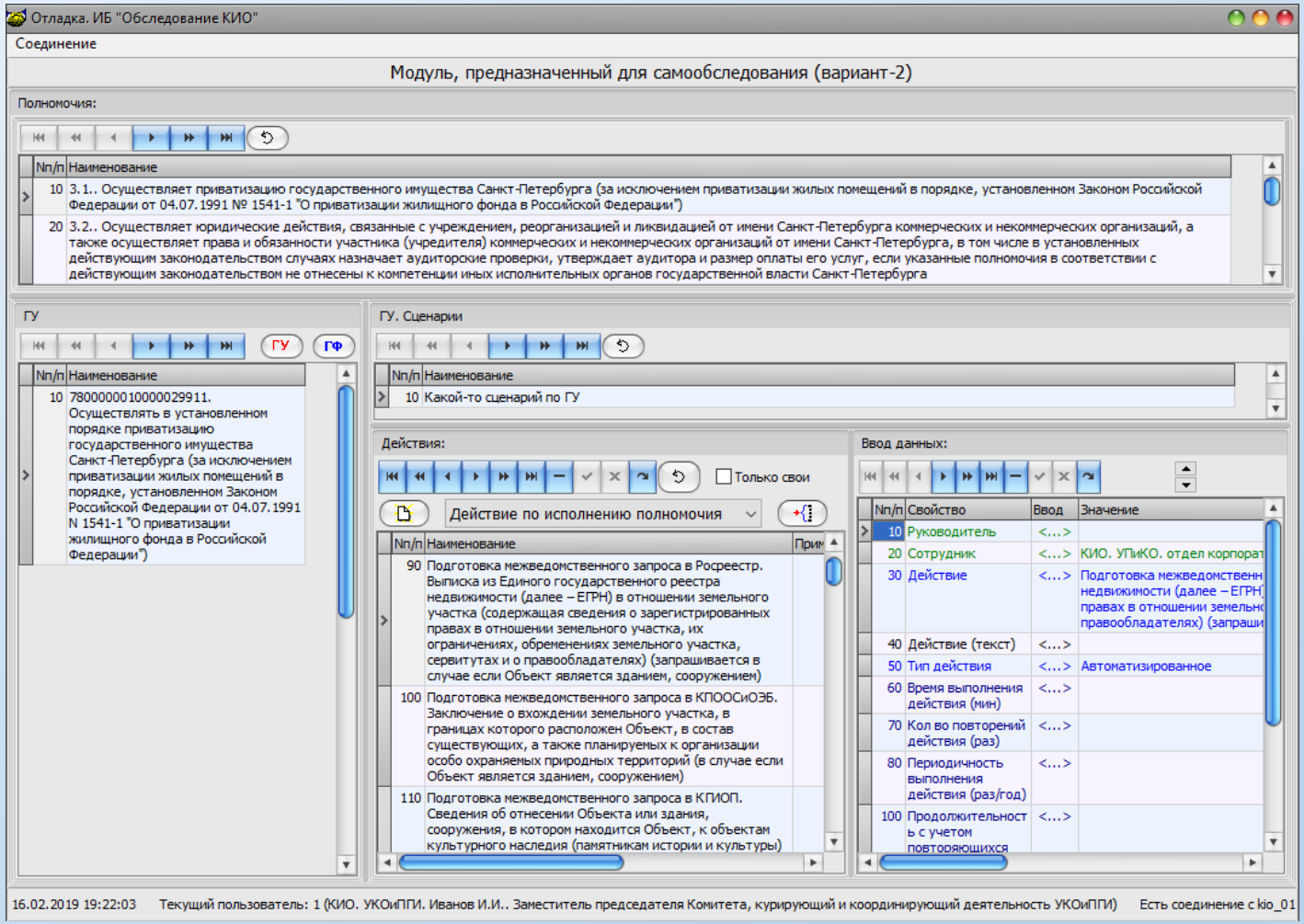
К главному меню…

# Метаданные. **Множество типов объектов**

## Хранение данных об информационной модели объектов данного типа.

## Реализация в нашей ИС.

Понятное дело, что GUI-форма, предназначенная для отображения и ввода данных может быть и не такой (как на соответствующих рисунках предыдущих слайдов).
А например, такая («жесткий» вариант GUI) см. рисунок ниже.




К главному меню…

Метаданные. **Множество типов объектов**

Хранение данных об информационной модели объектов данного типа.

Реализация в нашей ИС.

Следует отметить, что даже использование «жестких» вариантов GUI,
не уменьшает значимость основных преимуществ адаптивной ИС в целом.

Тем не менее, адаптивный и объектно-ориентированный подходы формирования ИС требует совершенного иного
(не стандартного) подхода при построении GUI (и это, пожалуй, одна из существенно знАчимых сложностей на всем проекте).

Это может быть, например, что-то вроде ууФ (унифицированная, управляемая GUI-форма).

Мы, кстати, начали прорабатывать возможность создания и применения ууФ (которая должна быть «одна и на все случаи жизни»).
Действующий макет ууФ у нас есть (см. рисунок на следующем слайде).

**Но там еще «копать и копать» (дай бог, чтобы не до «второго пришествия»)…**

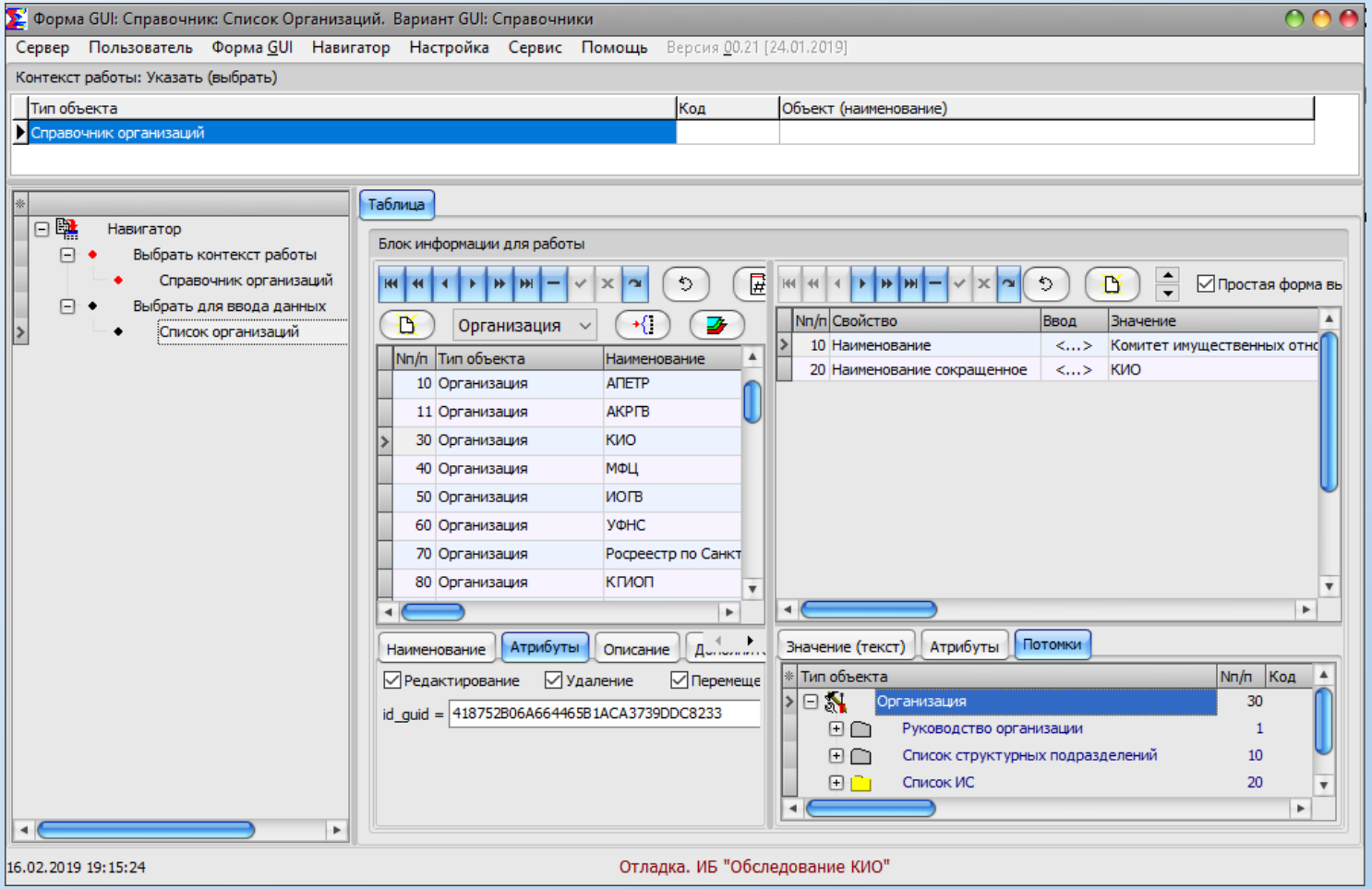
К главному меню…

# Метаданные. **Множество типов объектов**

Хранение данных об информационной модели объектов данного типа.

Реализация в нашей ИС.

Действующий макет ууФ (см. рисунок ниже).




К главному меню…

# Метаданные. **Множество типов объектов**

## Хранение данных об информационной модели объектов данного типа.

### Реализация в нашей ИС.

**ВАЖНО!**

1. Использование информационных моделей – факультативно. Т.е., - не хочешь – не создавай. Вводи все сам – ручками.
2. В любой момент времени Аналитик может изменить информационную модель так, как ему это необходимо.
3. Текущая информационная модель ОБЯЗАТЕЛЬНО и автоматически применяется для НОВЫХ объектов заданного типа.
4. Текущая информационная модель МОЖЕТ применяться для уже СУЩЕСТВУЮЩИХ объектов заданного типа.
   Но <u>только по команде</u> конечного пользователя.
5. Правила применения <u>текущей</u> информ. модели к уже <u>существующим</u> объектам включают ряд обязательных требований.
   В том числе, <u>исключение</u> возможности удаления уже существующей информации. Т.е., только «наращивание».
6. В ИС существует концептуальная возможность для конечного пользователя изменять информационную структуру существующего объекта (не путать с информационной моделью) по своему усмотрению.
7. В конечном итоге реализуется принцип:
   «Система должна предоставлять конечному пользователю возможность накапливать информацию по реальным объектам и их свойствам (из предметной области) с такой степенью детальности, «индивидуальности» и взаимосвязи между ними, с которой сам пользователь сочтет необходимым без концептуального ограничения со стороны Системы»

К главному меню…

# Метаданные. **Множество типов объектов**

## Хранение скрипта, предназначенного для автоматического вычисления наименований объектов данного типа.

Пример.

В обоих случаях (см. рисунки 1 и 2) информация об объектах введена корректно (на уровне свойств, см. рисунок 3).

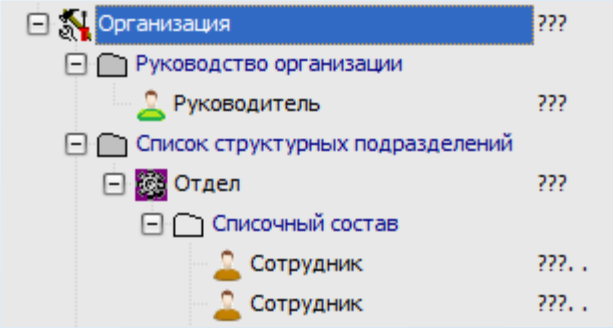



Рисунок 1

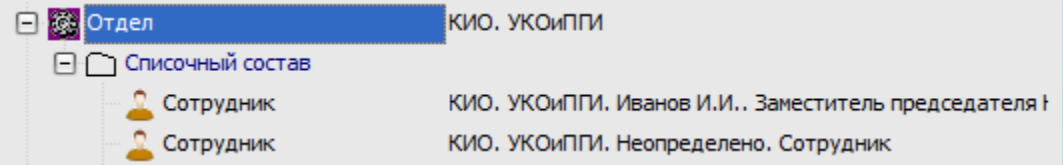



Рисунок 2

| Nп/п | Свойство | Значение |
|---|---|---|
| 10 | Должность | Заместитель председателя Комитета, курирующий и координирующий |
| 20 | Ф.И.О. | Неопределено |
| 30 | Ф.И.О. (сокращенно) | Иванов И.И. |
| 40 | Логин | 1 |

Рисунок 3

Но на рисунке 2 наименования есть, а на рисунке 1 - нет (что визуально, для человека, плохо).

Ну а что, в данном случае, адаптивная ИС (будучи, пока что, не ИИ) может сделать?
Она откуда может знать, как ей вычислять наименования объектов?

Для решения данной задачи есть два варианта:

1. Можно заполнить ручками (а значит, заставить пользователя вводить данные дважды).
2. Можно поручить это делать ИС (тогда пользователь вводит данные только один раз).

К главному меню…

# Метаданные. **Множество типов объектов**

## Хранение скрипта, предназначенного для автоматического вычисления наименований объектов данного типа.

### Реализация в нашей ИС.

Аналитик по определенным правилам (зная информационную модель предметной области, которую он сам же и сформировал) указывает ИС, как именно она должна автоматически произвести вычисление наименования объекта заданного типа на основании введенных конечным пользователем значений свойств (см. рисунок 1).

| Nп/п | Свойство | Значение |
|---|---|---|
| 10 | Должность | Заместитель председателя Комитета, курирующий и координирующий |
| 20 | Ф.И.О. | Неопределено |
| 30 | Ф.И.О. (сокращенно) | Иванов И.И. |
| 40 | Логин | 1 |

Рисунок 1

Скрипт выглядит так (см. рисунок 2).

```
<emp>~obj:Отдел_организации~.<nbsp>;<emp>~Ф_И_О_сокращенно~.<nbsp>;<emp>~Должность~<emp>;
```

Рисунок 2

Скажем так (мягко), что выглядит он весьма НеПриглядно, но:

1. Его интерпретация и выполнение производится на уровне триггеров PostgreSQL, а там сильно не попляшешь. Но зато неважно, какое решение используется (ВЕБ, Windows или иное). Разработчик об этом не заботится и результат везде будет одинаковым.
2. Должен быть нормальный, интерактивный инструментарий, который сформирует этот скрипт при вводе Аналитиком нормальных и понятных ему команд (у нас, кстати, такого инструмента нет и я все прописываю ручками, но одна из задач Тесонеро, как раз и есть - придумать такой инструмент по заданному мною алгоритму).

К главному меню…

# Метаданные. **Множество типов объектов**

## Системная поддержка одного из методов, реализующих механизм ссылочности объектов.

Реальные объекты предметной области могут проявлять (для наблюдателя) свои свойства в контексте других объектов, а также, в зависимости от условий исследования (наблюдения), воздействия и представления - по разному. При этом, **это один и тот же объект**.
Т.е., любой объект (являясь целостным, сложным и непрерывным в "пространстве-времени") может быть по разному представлен и "информационно усечен", в зависимости от задачи.
Для того, чтобы предоставить Пользователю возможность отобразить один и тот же объект в информационных структурах ИС определенным образом (в зависимости от контекста) и при этом "отследить" целостность объекта (как он существует в предметной области) - используется механизм ссылочности объектов.

В Системе предусмотрены два метода реализации механизма ссылочности объектов:

1. Ссылочные объекты (рассматриваются в данном разделе).
2. Ссылочные свойства (рассмотрены в разделе «Множество свойств»).

**Ссылочный объект** - это объект, ссылающийся на любой другой объект (включая и ссылочный, но исключая самого себя) и наследующий его определенные свойства (тип, наименование и т.д.).
Ссылочный объект может обладать своим набором свойств.

*Применительно к миварному пространству рискну предположить, что данный метод поддержки механизма ссылочности объектов, это (в частности) одна из составляющих такого понятия, как «отношение между объектами».*

*Примечание – в настоящий момент этот механизм в Системе отработан не достаточно полно (точнее – почти не отработан).*
*А на уровне GUI – практически вообще отсутствует.*
*Причины:*
*1. В текущей версии ИС пока что используется второй механизм ссылочности (на уровне ссылочных свойств).*
*2. Банальная причина – физическое отсутствие времени на проработку данного вопроса.*

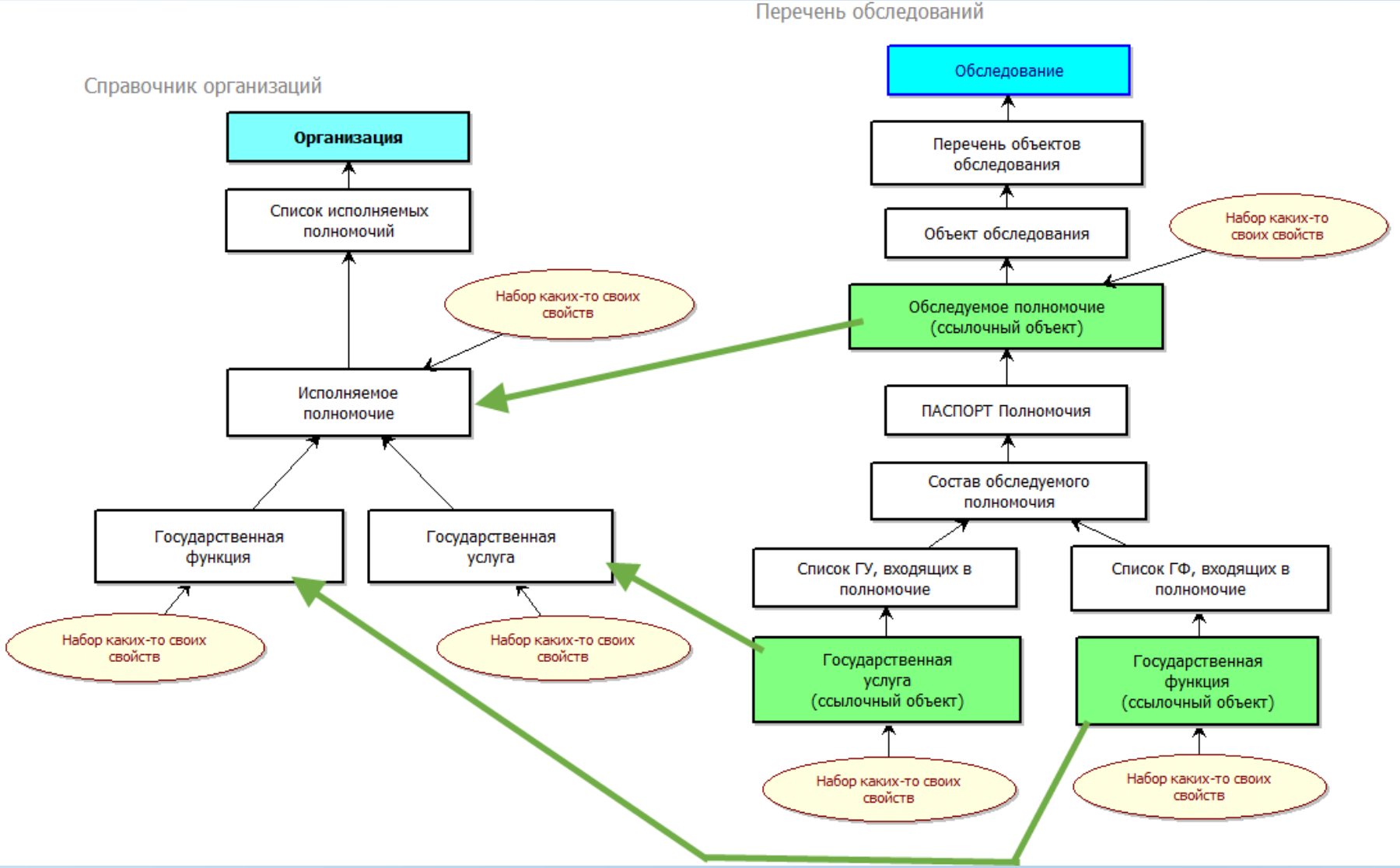
**Важно!** В настоящий момент в Системе приняты следующие правила:
объект-справочник может ссылаться только на другой объект-справочник;
объект может ссылаться как на любой объект-справочник, так и на любой другой объект.

К главному меню…

# Метаданные. **Множество типов объектов**

Системная поддержка одного из методов, реализующих механизм ссылочности объектов.

Ссылочные объекты. Иллюстрирующий пример (см. рисунок ниже).




К главному меню…

# Метаданные. **Множество типов объектов**

## Системная поддержка одного из методов, реализующих механизм ссылочности объектов.

### Важно-1

Данный метод механизма ссылочности (ссылочные объекты) – может быть дополнительно использован (при определенной доработке), как механизм наследования свойств объектов (из предметной области).
Но в текущей версии Системы это не реализовано (за ненадобностью).

### Важно-2

Не следует путать термин «объектно-ориентированность», как методологию программирования,
с термином «объектно-ориентированность» в контексте типовой информационной модели для предметной области.

К главному меню…

# Метаданные. **Множество типов объектов**

## Хранение скрипта для выбора объектов данного типа (при использовании механизма ссылочности, реализованного на уровне ссылочных объектов) с применением механизма предметных указателей.

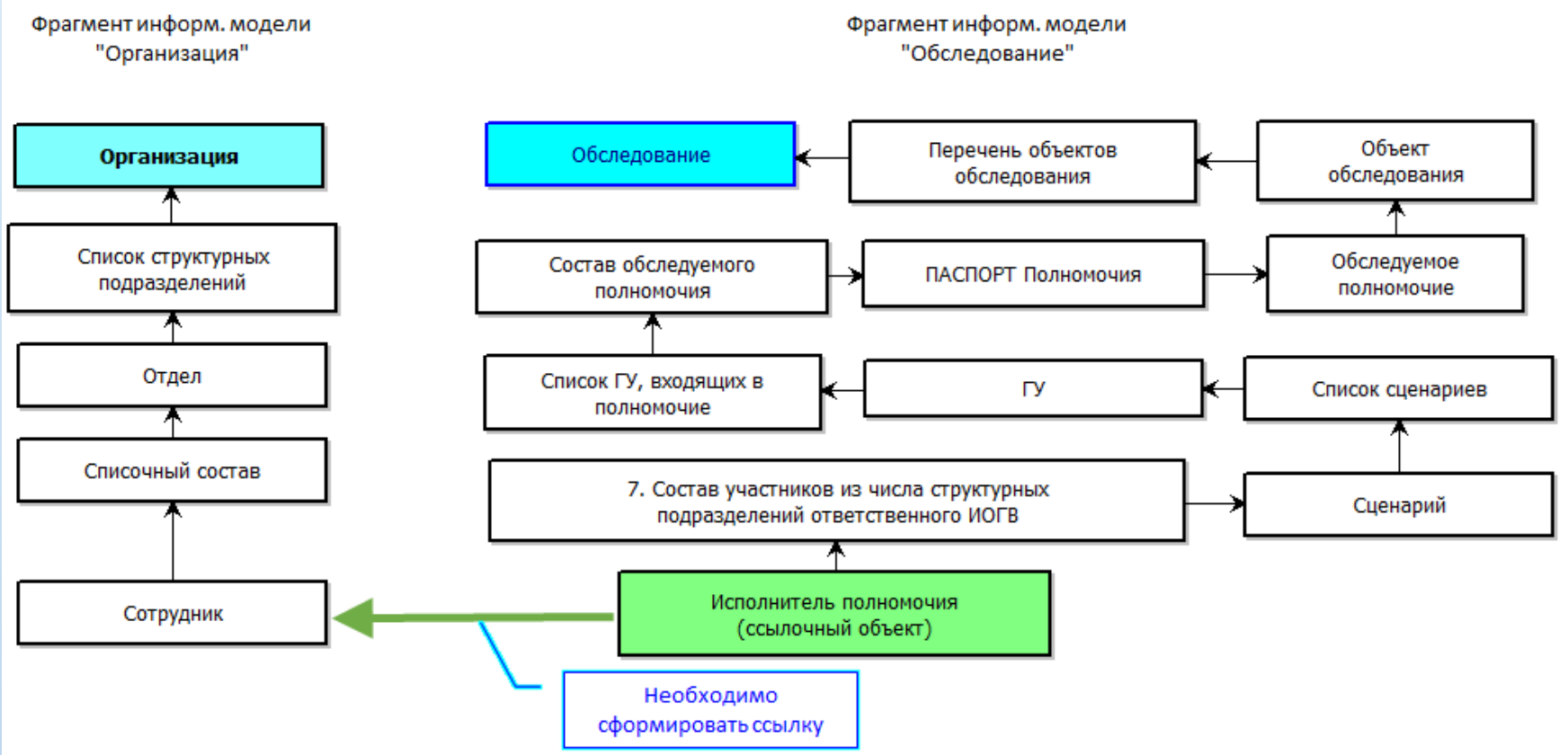
В процессе ввода данных возникает существенная проблема, связанная с интерактивным поиском нужного объекта для формирования ссылки на него.

Пример (см. рисунок ниже).

1. Есть некоторая информационная модель «Организация», где отражены данные об организациях и сотрудниках этих организаций.
2. Есть некоторая информационная модель «Обследование», где отражены данные об обследованиях и, в частности, об исполнителях полномочий.
3. Необходимо сформировать ссылку от объекта типа «Исполнитель полномочия» к объекту «Сотрудник».

**Важно!**

1. Сложность (включая и глубину иерархии) любой информационной модели заранее предсказать невозможно.
2. В процессе жизненного цикла любая информ. модель может измениться (тоже, в общем случае, непредсказуемо).
3. Объем данных (размер «дерева» объектов) может быть существенно большим.




К главному меню…

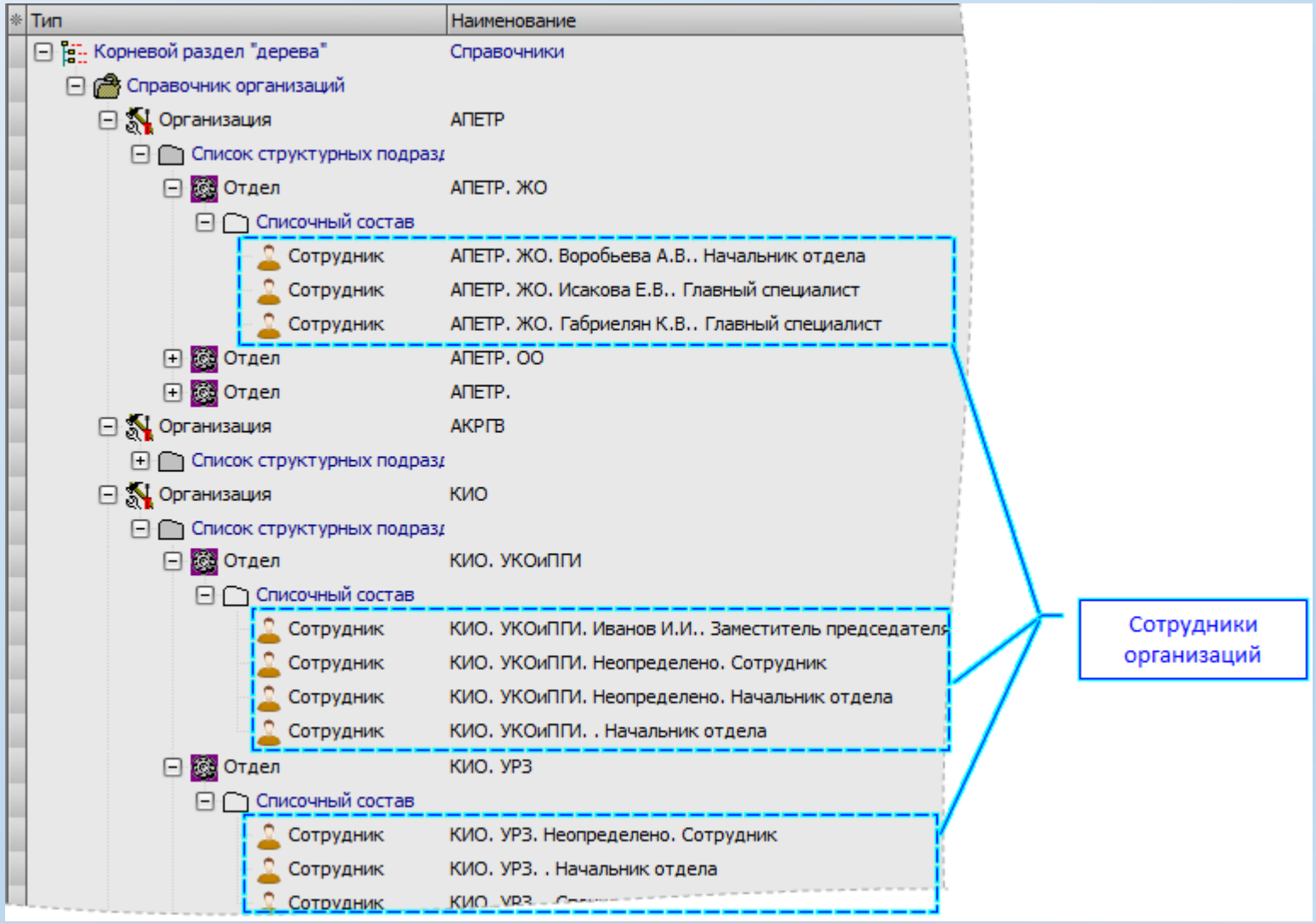
# Метаданные. **Множество типов объектов**

## Хранение скрипта для выбора объектов данного типа (при использовании механизма ссылочности, реализованного на уровне ссылочных объектов) с применением механизма предметных указателей.

Один из вариантов решения задачи.
Отобрать (отфильтровать) все объекты типа «Сотрудник» и, далее, сгруппировав их соответствующим образом (см. рисунок), выдать пользователю в виде GUI-формы для интерактивного выбора соответствующего объекта.

Очевидный и важный недостаток данного способа:
В случае большого объема данных (размера «дерева») время, затраченное на отбор (фильтрацию) данных, а также на отображение данных на GUI-форме может быть существенно большим (в общем случае – неприемлемо большим).

У нас, кстати, этот вариант реализован и даже проработан достаточно НеПлохо.




К главному меню…

# Метаданные. **Множество типов объектов**

## Хранение скрипта для выбора объектов данного типа (при использовании механизма ссылочности, реализованного на уровне ссылочных объектов) с применением механизма предметных указателей.

Другой вариант решения задачи.

1. Сформировать (по определенным правилам) перечень соответствующих предметных указателей (простые индексы), см. рисунок 1.
2. «Указать» ИС, что именно ей в дальнейшем надо делать (на уровне скрипта для выбора объектов данного типа), см. рисунок 2.
3. Далее, по сценарию (указанному в скрипте) N-е количество раз вызвать унифицированную и примитивную GUI-форму выбора объектов (предоставляя пользователю выбирать объекты поочередно), см. рисунок 3.

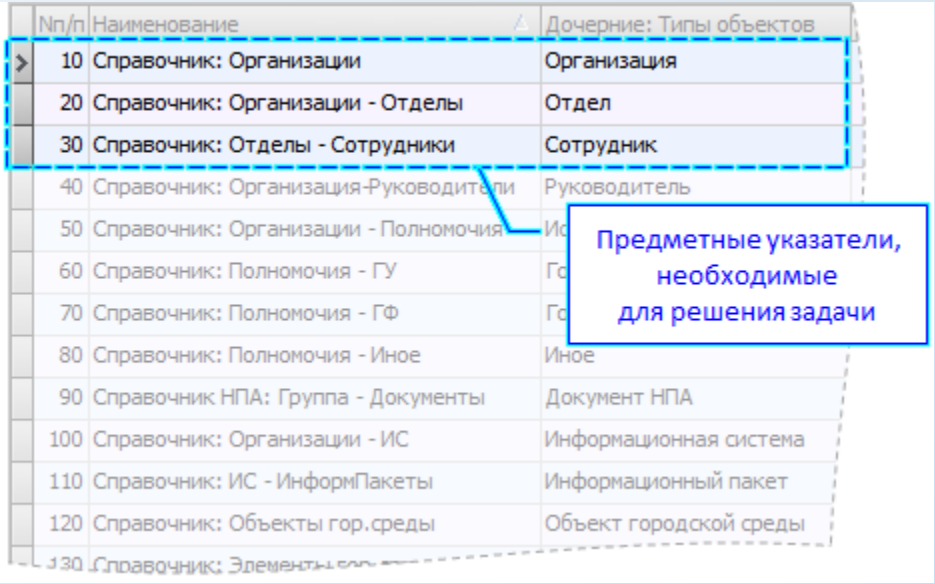



Рисунок 1.

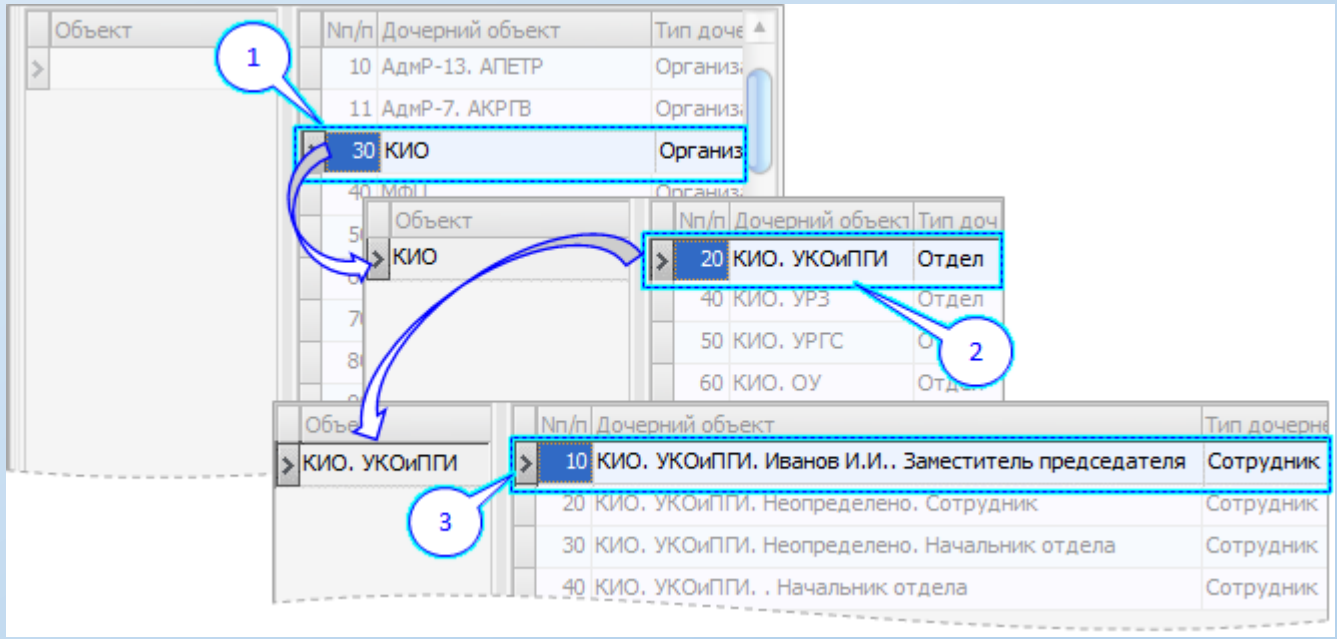



Рисунок 3.

```
{Справочник_Организации}
{Справочник_Организации_Отделы}
{Справочник_Отделы_Сотрудники}
```

Рисунок 2.

Очевидный плюс данного варианта в том, что предметные указатели (являясь, по сути, простыми индексами по ключевым полям таблиц БД dir.dir_tree и ws.obj_tree) позволяют существенно ускорить фильтрацию и поиск нужных объектов до приемлемого уровня.

К главному меню…

# Метаданные. **Множество типов объектов**

## Хранение скрипта для выбора объектов данного типа (при использовании механизма ссылочности, реализованного на уровне ссылочных объектов) с применением механизма предметных указателей.

И, кстати, этот механизм у нас НЕ до конца исчерпывающе и корректно отработан:
Причины:

1. Надо еще этот момент додумать, дощупать, и дореализовывать на макете.
   Потом тестить.
   И, если «что-то пойдет не так», то опять думать, щупать и т.д.
   А если совсем не пойдет, то придумывать что-то еще.
   Т.е., исследовать этот вопрос (в полном смысле этого слова).
2. Как и обычно – сейчас на это физически нет времени.

==========

В рамках формирования предметных указателей надо проверить, что выгоднее:

1. Производить актуализацию (индексирование) в реальном времени при добавлении/удалении каждого объекта (минус – каждый раз будет замедление работы на стороне клиента).
2. Производить актуализацию «ручками» (по команде конечного пользователя).
3. Производить актуализацию по таймеру (например, один раз в 10 – 30 секунд) в контексте отдельного процесса (программы).
4. Сейчас у нас реализованы 2-й и 3-й варианты (и, при этом, самый неэффективный метод – полное пересоздание предметных указателей).

==========

В рамках GUI : то, что у нас есть сейчас – крайне не оптимизировано.

К главному меню…

# Метаданные. **Множество свойств**

**Свойство** (объекта) – информационная сущность, определенным образом характеризующая информационный объект. Свойство не может быть «охарактеризовано» другими свойствами и в этом его коренное отличие от объекта.

Свойства объектов могут быть:
- системно предопределенные (всегда присутствуют у любого объекта);
- определенные пользователем (задаются конечным пользователем).

**Множество свойств** – перечень всех свойств, «известных» Системе.
Если появляются новые (определенные пользователем) свойства, которые будут применяться в дальнейшем, то они должны быть сначала интерактивно внесены конечным пользователем в «множество свойств» по определенным правилам.
Количество определенных пользователем свойств концептуально не ограничено.

См., также, здесь…

# Метаданные. Множество свойств. **Системно предопределенные свойства**

Перечень системно предопределенных свойств приведен в таблице ниже.

| Свойство | Поле таблицы БД | Примечание |
|---|---|---|
| Уникальный идентификатор объекта (или объекта-справочника) | id_guid | Задается в клиентском ПО или, в противном случае, вычисляется автоматически в триггере «before insert». Значение не меняется в течение всего жизненного цикла объекта |
| Уникальный идентификатор типа объекта (текущий, с учетом ссылочности) | id_type_guid | Автоматически наследуется от того объекта, на который указывает ссылка в id_link_guid (если это ссылочный объект) в триггере.<br>В противном случае: id_type_guid = id_type_orig_guid (в триггере) |
| Уникальный идентификатор типа объекта (первоначальный, назначенный при создании объекта) | id_type_orig_guid | Задается в клиентском ПО. Значение не меняется в течение всего жизненного циакла объекта.<br>**Важно!** В текущей версии принято, что при создании нового объекта его тип (первоначальный) должен быть задан в клиенском ПО в поле id_type_guid (в триггере «before insert» выполняется id_type_orig_guid = id_type_guid до иных действий) |
| Уникальный идентификатор «родителя» объекта | id_parent_guid | Задается в клиентском ПО |
| Признак абстрактности объекта (true - объект абстрактный) | its_abstract | Автоматически наследуется (в триггере) от типа объекта: id_type_guid |
| Признак ссылочности объекта (true - объект ссылочный) | its_link | Автоматически вычисляется (в триггере) по значению id_link_guid |
| Тип ссылки | link_type | Автоматически вычисляется в зависимости от таблицы (dir.dir_tree или ws.obj_tree) в триггере.<br>Возможные значения для объектов-справочников:<br>• отсутствует - см. public.lt_is_all();<br>• ссылка на объект-справочник - см. public.lt_is_dir().<br>Возможные значения для объектов:<br>• отсутствует - см. public.lt_is_all();<br>• ссылка на объект-справочник - см. public.lt_is_dir();<br>• ссылка на объект - см. public.lt_is_obj() |
| Ссылка на объект (или объект-справочник). Уникальный идентификатор | id_link_guid | Если это НЕ ссылочный объект, то пустая строка |

К главному меню…

# Метаданные. Множество свойств. **Системно предопределенные свойства**

Продолжение таблицы…

| Свойство | Поле таблицы БД | Примечание |
|---|---|---|
| Наименование полное объекта | name_max | Если объект ссылочный, то автоматически наследуется (в триггере) от того объекта, на который указывает ссылка в id_link_guid |
| Наименование сокращенное объекта | name_min | Если объект ссылочный, то автоматически наследуется (в триггере) от того объекта, на который указывает ссылка в id_link_guid |
| Комментарий к объекту | note | |
| Детальное описание объекта | info | |
| Пользовательский код объекта | code | Если объект ссылочный, то автоматически наследуется (в триггере) от того объекта, на который указывает ссылка в id_link_guid |
| Мнемокод объекта | mnemocode | Задается в клиентском ПО или, в противном случае, вычисляется автоматически в триггере «before insert» |
| Уровень в иерархии | level_this | Автоматически наследуется (в триггере) от «родителя» |
| Порядковый номер | npp | Задается в клиентском ПО или, в противном случае, вычисляется автоматически в триггере «before insert» в контексте «родителя» (id_parent_guid) |

К главному меню…

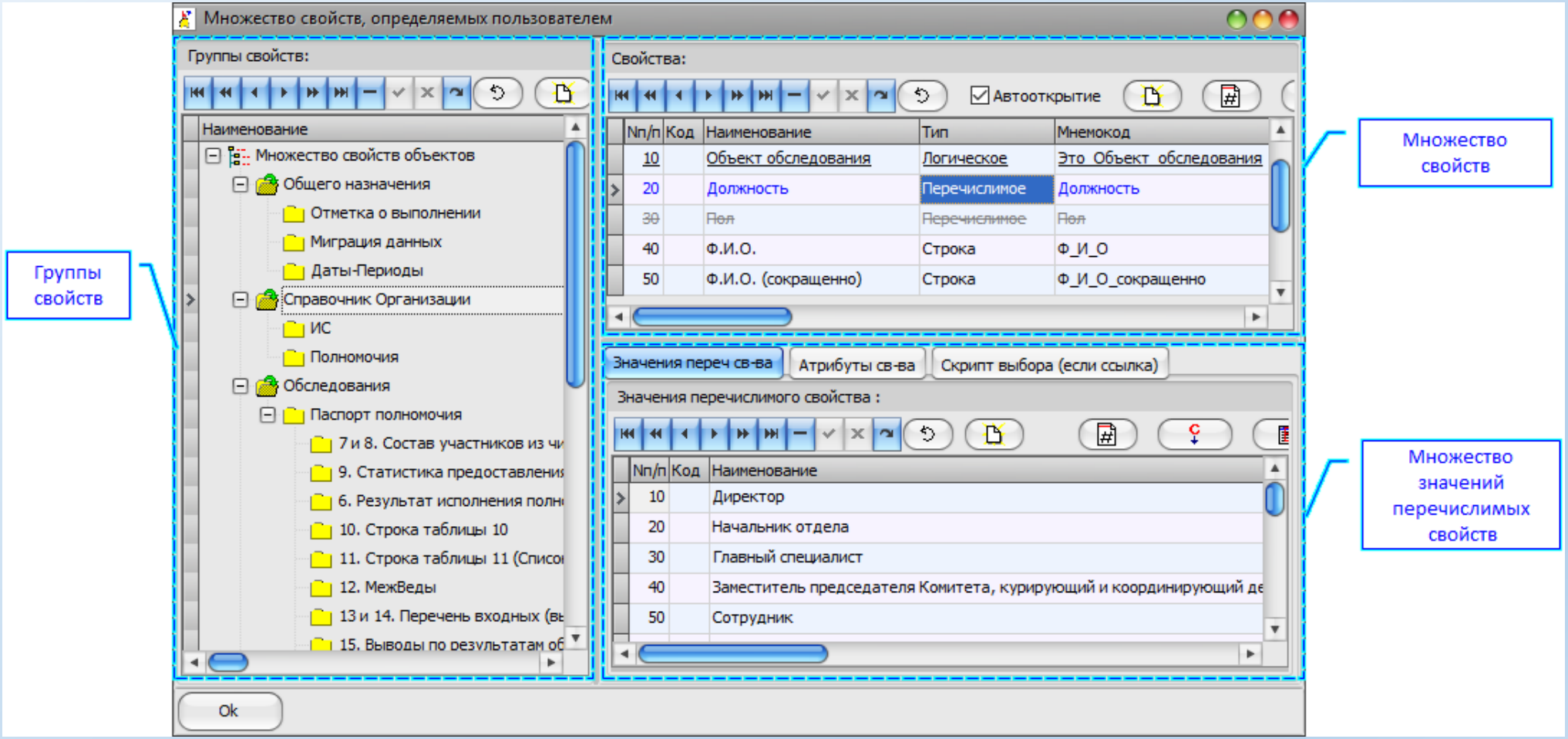
# Метаданные. **Множество свойств, определяемых пользователем**

## Практическая реализация «множества свойств»

На уровне PostgreSQL-сервера:

1. Множество свойств (имя таблицы БД: «dir.props_list»).
2. Множество значений перечислимых свойств (имя таблицы БД: «dir.props_enum_vals»).
3. Группы свойств (имя таблицы БД: «dir.props_grp»).
4. N-е количество ХФ.

На уровне GUI (см. рисунок):




К главному меню…

# Метаданные. **Множество свойств, определяемых пользователем**

Основное назначение (в контексте объектной модели предметной области):

1. Строгая и однозначная классификация свойств объектов по видовому признаку (вид свойства).
2. Системная поддержка одного из методов, реализующих механизм ссылочности объектов (ссылочные свойства).
3. Хранение скрипта для выбора объектов (при использовании механизма ссылочности, реализованного на уровне ссылочных свойств) с применением механизма предметных указателей.

**Важно!**

1. В общем случае, множество типов объектов и множество свойств (определяемых пользователем) – концептуально никак не связаны друг с другом.
2. В процессе формирования информационной модели предметной области, Аналитик принимает решение, какие именно свойства (из «множества свойств») будут использованы при создании новых информационных объектов (по умолчанию).
3. Конечному пользователю предоставлена возможность добавлять или удалять свойства (из «множества свойств») для любого информационного объекта.
4. Нет обязательных требований, чтобы все свойства были «использованы» в процессе ввода информации о реальных объектах из предметной области.
5. Свойства (из «множества свойств») используются, также, при настройке процессов миграции данных.

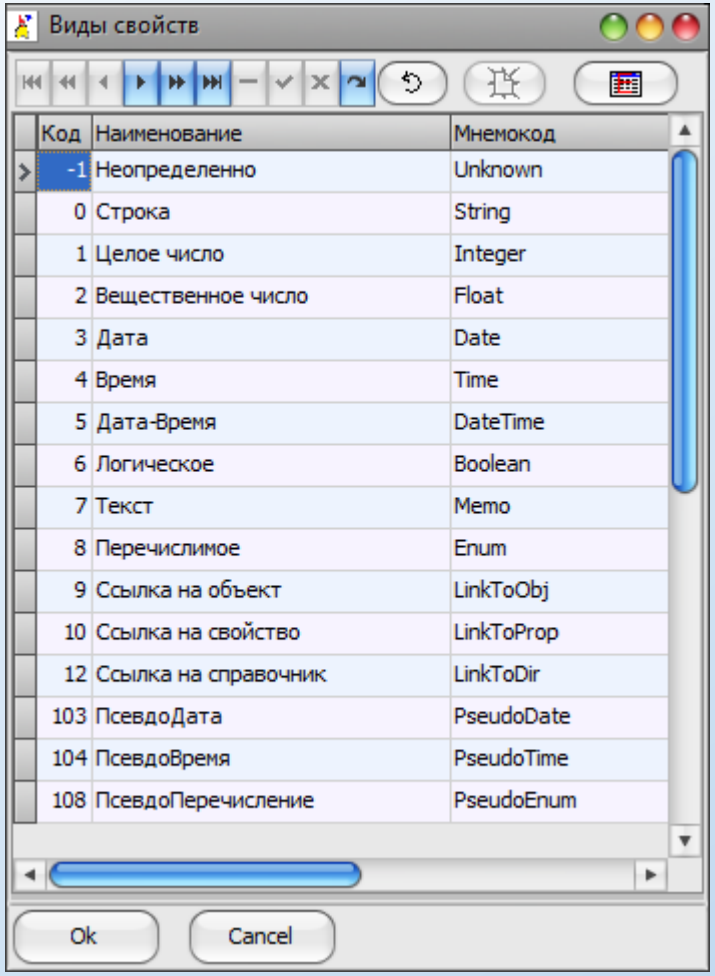
*Примечание – процесс миграции данных у нас реализован пока что только в части настроек и то, на уровне «декларации намерений».*
*Т.е., все осталось на уровне «мысленного эксперимента».*

К главному меню…

# Метаданные. **Множество свойств, определяемых пользователем**

## Виды свойств

Перечень допустимых видов свойств, определяемых пользователем, приведен на рисунке ниже.




| Код | Наименование | Мнемокод |
|---|---|---|
| -1 | Неопределенно | Unknown |
| 0 | Строка | String |
| 1 | Целое число | Integer |
| 2 | Вещественное число | Float |
| 3 | Дата | Date |
| 4 | Время | Time |
| 5 | Дата-Время | DateTime |
| 6 | Логическое | Boolean |
| 7 | Текст | Memo |
| 8 | Перечислимое | Enum |
| 9 | Ссылка на объект | LinkToObj |
| 10 | Ссылка на свойство | LinkToProp |
| 12 | Ссылка на справочник | LinkToDir |
| 103 | ПсевдоДата | PseudoDate |
| 104 | ПсевдоВремя | PseudoTime |
| 108 | ПсевдоПеречисление | PseudoEnum |

Когда конечный пользователь вносит новое свойство в Систему, то **вид свойства** обязательно должен быть выбран из указанного **перечня** (см. рисунок).

Исключение составляет вид свойства «Неопределенно» (код = -1).
Его использовать (в рамках текущей реализации) – не представляется возможным (установлен запрет на уровне ядра Системы).

К главному меню…

# Метаданные. **Множество свойств, определяемых пользователем**

## Виды свойств

Перечень допустимых видов свойств, определяемых пользователем, приведен в таблице ниже.

| Код | Мнемокод | Наименование | ХФ, возвращающие соответствующие значения | Допустимые значения | Примечание |
|---|---|---|---|---|---|
| -1 | Unknown | Неопределенно | public.pt_is_unknown() | - | Резерв. В данной версии это значение НЕ допустимо |
| 0 | String | Строка | public.pt_is_str() | Пустая строка или строка длиной не более, чем 4800 символов (домен public.t_propval) | |
| 1 | Integer | Целое число | public.pt_is_int() | Пустая строка или строка символов, корректно конвертируемая в тип integer | |
| 2 | Float | Вещественное число | public.pt_is_float() | Пустая строка или строка символов, корректно конвертируемая в вещественное значение (double precision) | |
| 3 | Date | Дата | public.pt_is_date() | Пустая строка или строка символов, корректно интерпретирумая, как дата "День Месяц Год" | |
| 4 | Time | Время | public.pt_is_time() | Пустая строка или строка символов, корректно интерпретирумая, как время "Часы Минуты" или "Часы Минуты Секунды" | |
| 5 | DateTime | Дата-Время | public.pt_is_datetime() | Пустая строка или строка символов, корректно конвертируемая в тип datetime | |
| 6 | Boolean | Логическое | public.pt_is_bool() | Пустая строка или строка символов, корректно конвертируемая в тип boolean | |

К главному меню…

# Метаданные. **Множество свойств, определяемых пользователем**

## Виды свойств

Продолжение таблицы…

| Код | Мнемокод | Наименование | ХФ, возвращающие соответствующие значения | Допустимые значения | Примечание |
|---|---|---|---|---|---|
| 7 | Memo | Текст | public.pt_is_memo() | Текст | Для того, чтобы значению свойства этого типа было автоматически (в триггере) присвоено NULL, следует на уровне клиентского ПО присвоить ему (поле prop_val_memo) строковое значение {[(SET_AS_NULL)]} |
| 8 | Enum | Перечислимое | public.pt_is_enum() | Пустая строка или корректное значение id_guid из таблицы БД dir.props_enum_vals | Перечислимое свойство можно рассматривать, также, как простейший классификатор |
| 9 | LinkToObj | Ссылка на объект | public.pt_is_linktoobj() | Пустая строка или корректное значение id_guid из таблицы ws.obj_tree | Исключая ссылку на «самое себя» |
| 10 | LinkToProp | Ссылка на свойство | public.pt_is_linktoprop() | Пустая строка или корректное значение id_guid из таблиц dir.dir_props или ws.obj_props | В данной версии не используется |
| 12 | LinkToDir | Ссылка на справочник | public.pt_is_linktodir() | Пустая строка или корректное значение id_guid из таблицы dir.dir_tree | Ссылка на объект-справочник. Исключая ссылку на «самое себя» |

К главному меню…

# Метаданные. **Множество свойств, определяемых пользователем**

## Виды свойств

Продолжение таблицы…

| Код | Мнемокод | Наименование | ХФ, возвращающие соответствующие значения | Допустимые значения | Примечание |
|---|---|---|---|---|---|
| 103 | PseudoDate | ПсевдоДата | public.pt_is_pseudodate() | Пустая строка или строка длиной не более, чем 4800 символов (домен public.t_propval) | Если значение данного свойства может быть интепретировано, как корректная дата в формате "День Месяц Год", то оно (значение) нормализуется (приводится) к формату ДД.ММ.ГГГГ.<br>В противном случае - значение остается "как есть" (as is).<br>Не реализовано в данной версии |
| 104 | PseudoTime | ПсевдоВремя | public.pt_is_pseudotime() | Пустая строка или строка длиной не более, чем 4800 символов (домен public.t_propval) | Если значение данного свойства может быть интепретировано, как корректное время в формате "Часы Минуты" или "Часы Минуты Секунды", то оно (значение) нормализуется (приводится) к формату ЧЧ:ММ или ЧЧ:ММ:СС, соответственно.<br>В противном случае - значение остается "как есть" (as is).<br>Не реализовано в данной версии |
| 108 | PseudoEnum | ПсевдоПеречисление | public.pt_is_pseudoenum() | Пустая строка или строка длиной не более, чем 4800 символов (домен public.t_propval) | Если значение данного свойства может быть интепретировано, как корректное значение (id_guid) какого-либо значения данного свойства (как перечислимого, см., также, таблицу БД dir.props_enum_vals) из множества свойств, то ХФ public.dirobj_prop_val_vr_get() возвращает значение перечислимого свойства (из таблицы БД dir.props_enum_vals).<br>В противном случае - значение остается "как есть" (as is).<br>Не реализовано в данной версии |

К главному меню…

# Метаданные. **Множество свойств, определяемых пользователем**

## Ссылочные свойства

Виды свойств, предназначенные для формирования ссылок приведены ниже:

**LinkToObj** - ссылка на объект (таблица БД ws.obj_tree);

**LinkToDir** - ссылка на объект-справочник (таблица БД dir.dir_tree);

**LinkToProp** - ссылка на другое свойство (в текущей реализации не используется и, соответственно, не проработано в должной мере).

**Важно!**

Основное отличие метода ссылочных свойств от метода ссылочных объектов в том, что ссылочный объект может ссылаться только на один объект, в то время как использование ссылочных свойств позволяет сформировать множество ссылок в контексте конкретного объекта. Количество таких ссылок концептуально не ограничено.

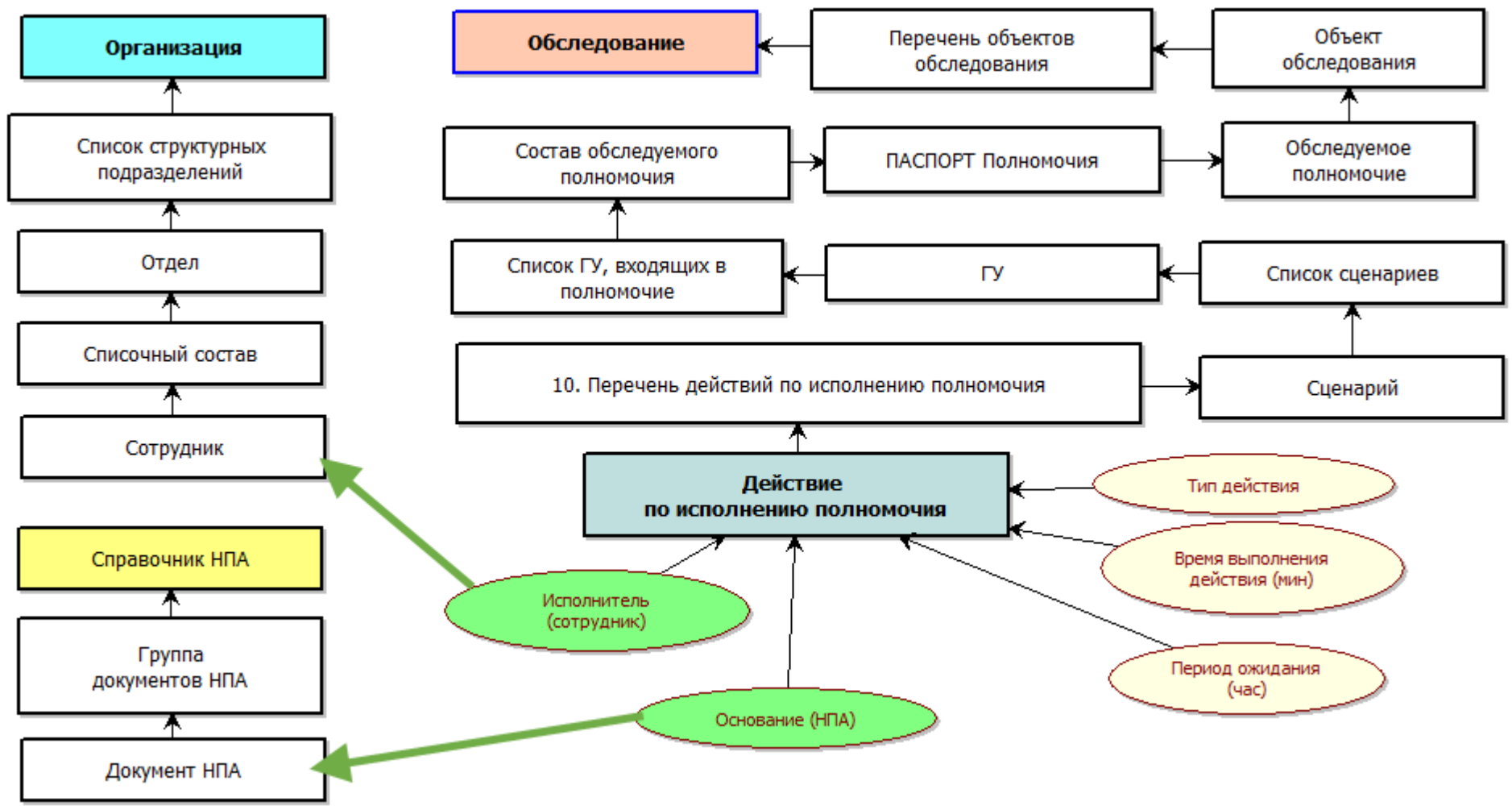
Далее приведен пример, иллюстрирующий применение ссылочных свойств.

К главному меню…

# Метаданные. **Множество свойств, определяемых пользователем**

## Ссылочные свойства

Иллюстрирующий (гипотетический) пример




Зеленым фоном отмечены ссылочные свойства объекта «Действие по исполнению полномочия».
Зелеными стрелками отмечены ссылки на соответствующие объекты.

К главному меню…

# Метаданные. **Множество свойств, определяемых пользователем**

## Скрипт для выбора объектов (формирование ссылки) с применением механизма предметных указателей

Применительно к данному разделу, актуальна информация, приведенная здесь, здесь и здесь.

Повторять ее здесь не имеет смысла.

К главному меню…

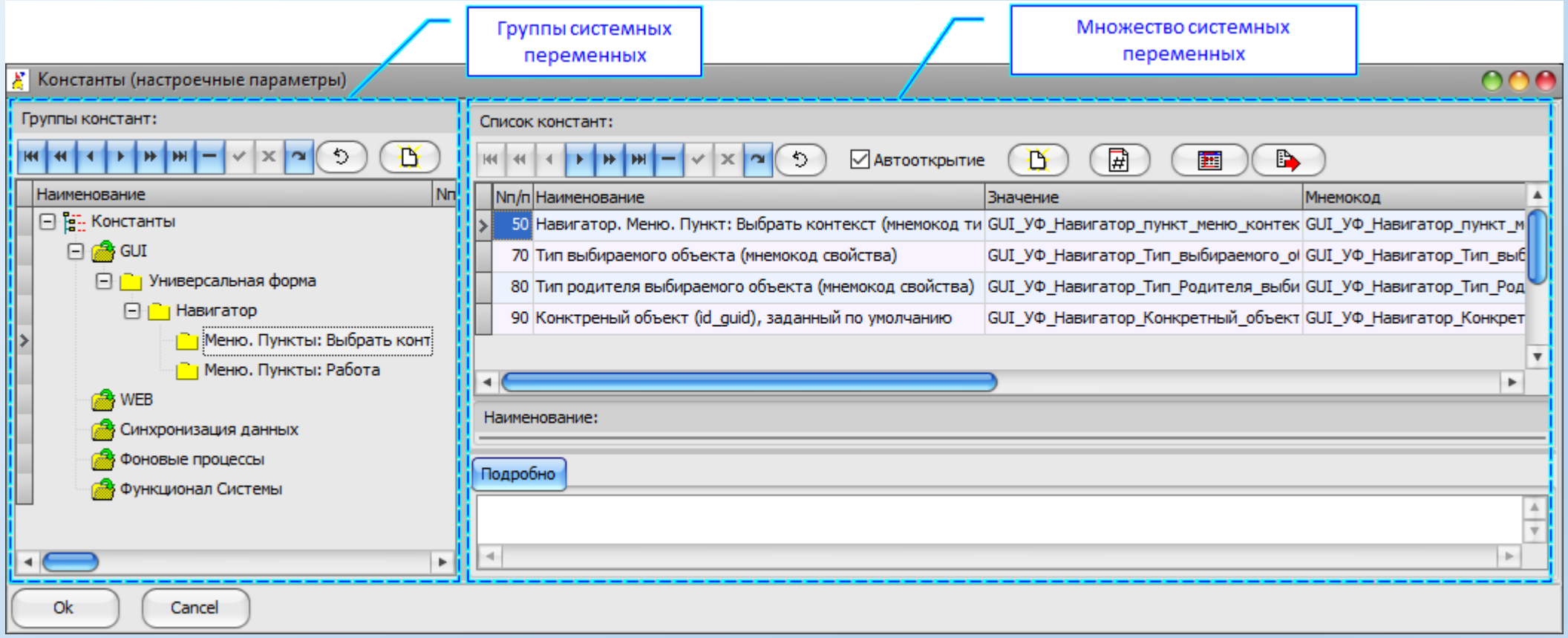
# **Системные переменные.** Практическая реализация

**Системная переменная** - информационная именованная сущность, используемая определенным образом для настройки Системы.

На уровне PostgreSQL-сервера:

1. Множество системных переменных (имя таблицы БД: «dir.const_list»).
2. Группы системных переменных (имя таблицы БД: «dir.const_grp»).
3. N-е количество ХФ.

На уровне GUI (см. рисунок):




К главному меню…

# Предметные указатели

**Аксиома.**
В общем случае, «узко-заточенная» ИС (разработанная под конкретную прикладную задачу) по быстродействию поискового механизма превзойдет адаптивную (или претендующую на ее роль) ИС, используемую для этой же задачи.

Предметный указатель (по аналогии с СИБИД) - перечень объектов (потомков) заданного типа, определенных в контексте их "предков" (также объектов заданного типа).
Предметный указатель содержит уникальные идентификаторы соответствующих объектов.

Предназначен для ускорения и упрощения операций, связанных с :

1. Поиском информации.
2. Интерактивным вводом информации (поиск и выбор объекта) при формировании, в частности, ссылочных объектов и свойств.

Предметные указатели формируются (при необходимости) пользователями в процессе формирования (уточнения) информационных моделей.

Допустима ассоциация предметного указателя с «привычным» индексом (в БД).

К главному меню…

# Предметные указатели. **Поиск информации**

Сначала – вводный пример.

1. Есть данность: некая информационная модель, сформированная аналитиком (см. рисунок 1).
2. Есть задача: периодически (существенно часто) составлять списки **<u>всех</u>** сотрудников какой-либо организации вне зависимости от отделов (например, для КИО, см. рисунок 2).

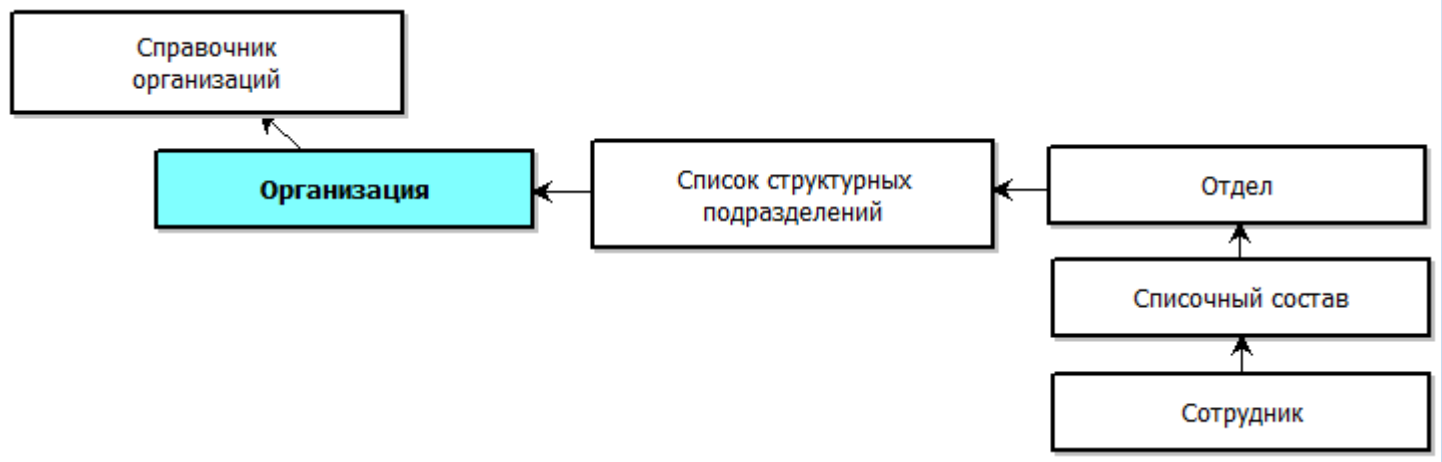



Рисунок 1

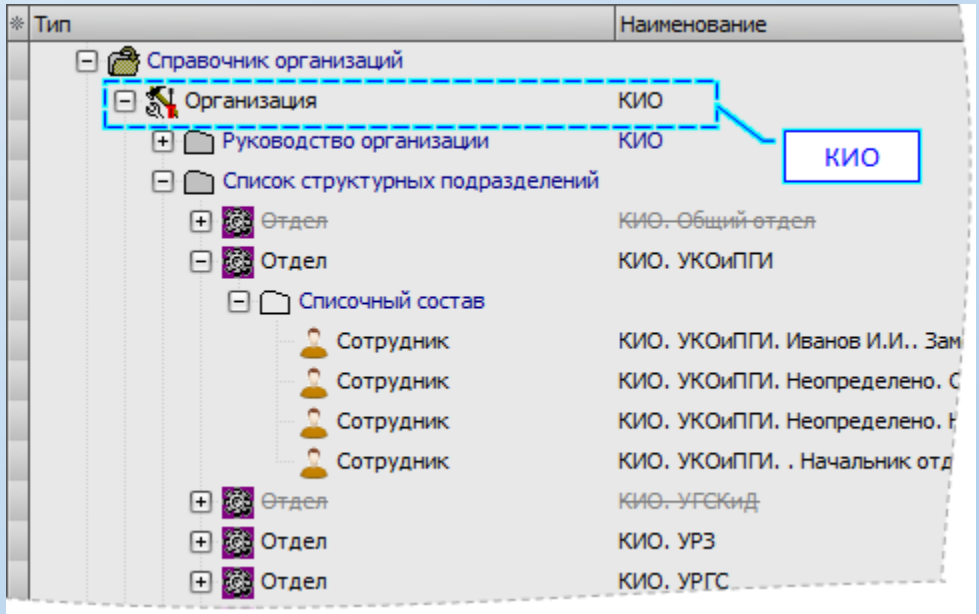



Рисунок 2

Возможных способов решения – достаточно много.
Например, некоторые из них:
пошаговые действия: сначала найти все отделы, а затем – по каждому отделу отобрать сотрудников;
сложный, стандартный SQL-запрос (возможно, с использованием JOIN);
простой, специализированный SQL-запрос (предусмотренный в Системе для таких случаев).

Но в случае большого объема данных, сложности информационной модели и критичности времени выполнения задачи – вышеперечисленные способы могут оказаться неприемлемыми.

К главному меню…

# Предметные указатели. **Поиск информации**

Для таких случаев может быть использован механизм предметных указателей.

Аналитик интерактивно создает соответствующий предметный указатель.
В нашем случае (см. пример выше) это будет выглядеть так (фрагмент визуального представления, см. рисунок ниже).

| Тип родитель | Родительский объект | Дочерний объект | Тип дочернего |
|---|---|---|---|
| Организация | КИО | КИО. УПиКО. Отдел корпоративных | Сотрудник |
| Организация | КИО | КИО. УРГС. . Начальник отдела | Сотрудник |
| Организация | КИО | КИО. УРЗ. . Специалист | Сотрудник |
| Организация | КИО | КИО. УКОиППИ. Неопределен | Сотрудник |
| Организация | КИО | КИО. УПСОН. . Начальник отдела | Сотрудник |
| Организация | КИО | КИО. УКОиППИ. . Начальник отдела | Сотрудник |
| Организация | КИО | КИО. УРГС. . Главный | Сотрудник |
| Организация | КИО | КИО. УПиКО. Отдел приватизации (позже | Сотрудник |
| Организация | Городская комиссия | Городская комиссия.. . Сотрудник | Сотрудник |
| Организация | КПООСиОЭБ | КПООСиОЭБ.. . Сотрудник | Сотрудник |
| Организация | ФГБУ "ФКП | ФГБУ "ФКП Росреестра". Сотрудник | Сотрудник |
| Организация | ЖК | ЖК. ОКН. . | Сотрудник |
| Организация | ЖК | ЖК. САП. . | Сотрудник |
| Организация | ЖК | ЖК. ОРЖФ. . | Сотрудник |
| Организация | ЖК | ЖК. ОНФОЖ. . | Сотрудник |
| Организация | ЖК | ЖК. ОСГЛКУЖХ | |

Т.е., это уже простая, соответствующим образом проиндексированная таблица в Б.Д.
Где в соответствие приведены следующие пары объектов соответствующих типов:

```
УникальныйКод Сотрудника - УникальныйКод Организации
```

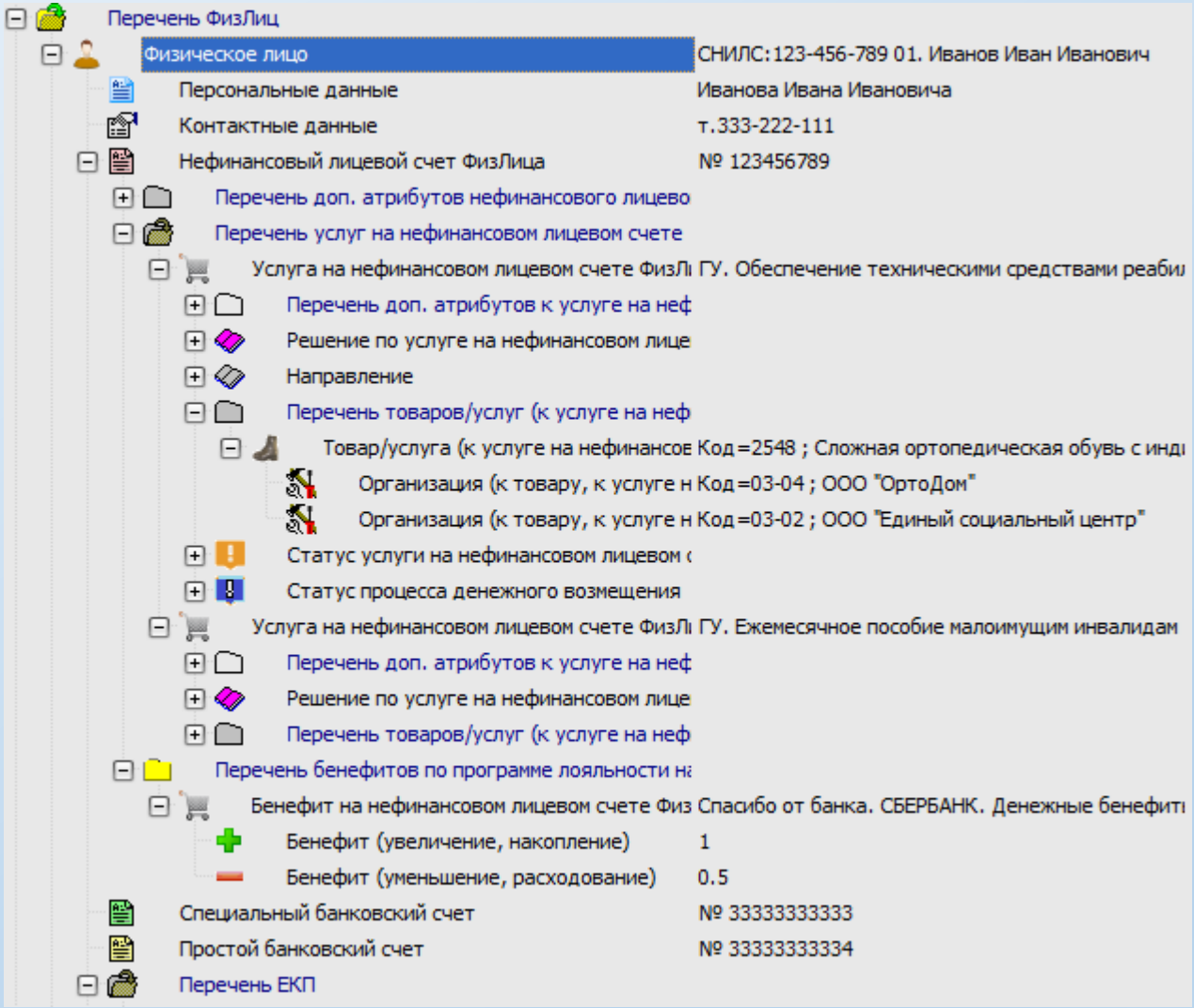
К главному меню…

# Предметные указатели. **Поиск информации**

Как еще один пример актуальности данной проблемы (поиск информации в адаптивной ИС) – еще один иллюстрирующий пример (в контексте очень нам всем «близкой и знакомой» темы - ЕКП).

**<u>Гипотетический пример</u>**: есть сформированная Аналитиком некоторая информационная модель (см. рисунки ниже).

**Начало**




К главному меню…

# Предметные указатели. **Поиск информации**

## Продолжение и окончание…

Очевидно, что при такой сложности информационной модели и большом объеме данных – без предметных указателей (или подобного механизма) не обойтись.


К главному меню…

# Предметные указатели. **Актуализация**

**Важно!**

В рамках формирования предметных указателей у нас (в моем лице) все еще нет четкого понимания, что выгоднее:

1. Производить актуализацию (индексирование) в реальном времени при добавлении/удалении каждого объекта.
   Минус – каждый раз будет замедление работы на стороне клиента.
2. Производить актуализацию (индексирование) по таймеру (например, один раз в 10 – 30 секунд) при использовании отдельного процесса (программы). Минус – должен где-то «крутиться» дополнительный процесс (программный модуль) и, кроме этого, будет некоторое запаздывание в доступности соответствующих объектов в контексте предметных указателей.

Т.е., этот момент надо еще прорабатывать, тестировать, сравнивать, а потом принимать окончательное решение.

Сейчас у нас реализованы два способа актуализации (индексирования):

1. «Ручками». Пользователь нажимает на соответствующую кнопку (или выбирает соответствующий пункт меню).
2. По таймеру. Для этого есть специальный программный модуль (но слабо тестированный, а вернее – вообще не было реального тестирования).

И, кроме этого, мы (в настоящий момент) используем самый неэффективный (но самый надежный) метод актуализации – полное пересоздание каждого предметного указателя.
Причина банальна – физически не было времени на проработку вопроса. Так что реализовал по минимуму…

К главному меню…

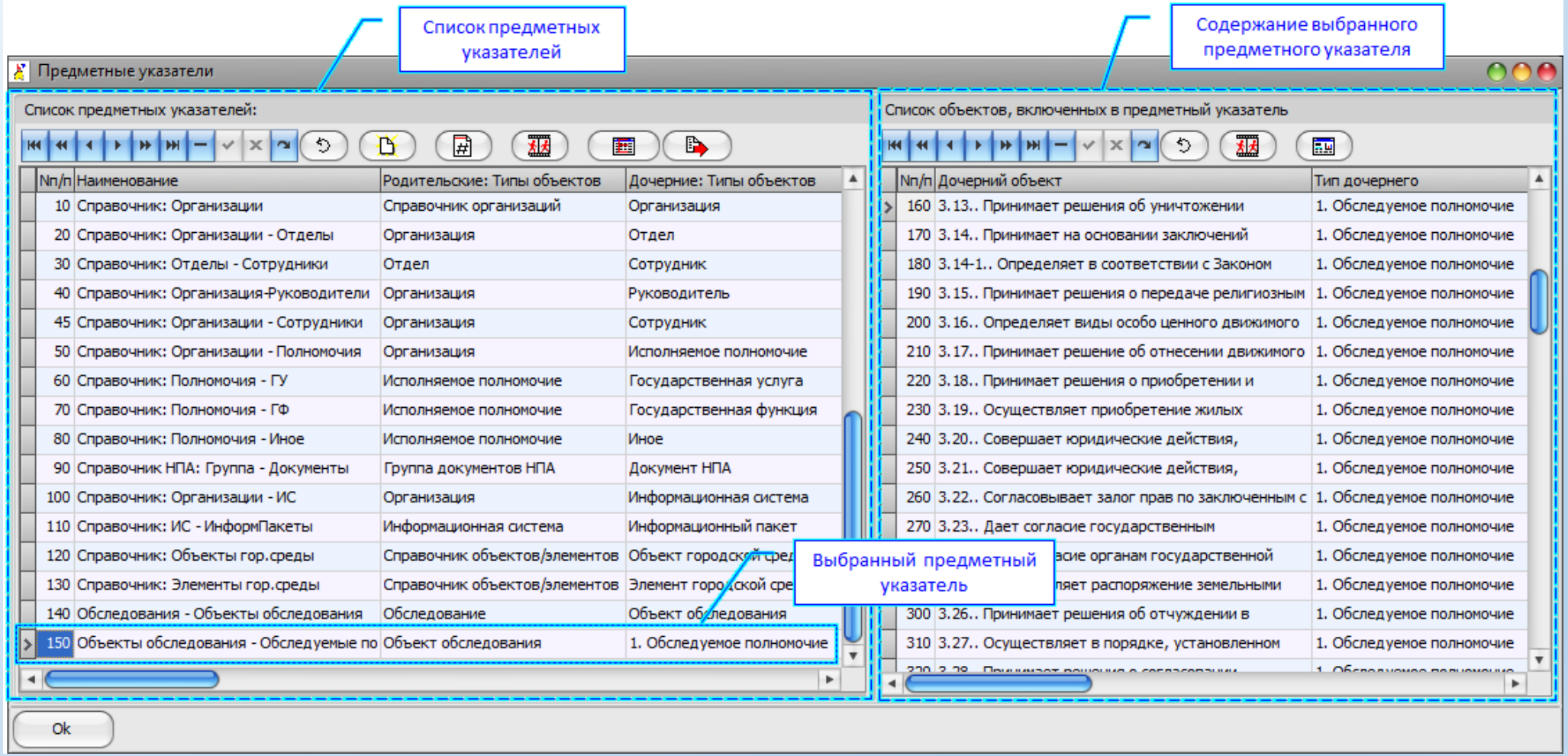
Существенные проблемы, связанные с интерактивным поиском и выбором объектов (а также механизм использования предметных указателей для решения этих проблем) рассмотрены здесь и здесь.

# Предметные указатели. **Практическая реализация**

На уровне PostgreSQL-сервера:

1. Множество предметных указателей (имя таблицы БД: «si.si_list»).
2. Содержание предметных указателей – пары объектов соответствующих типов (имя таблицы БД: «si.obj_list»).
3. N-е количество соответствующих ХФ.

На уровне GUI (см. рисунок):




К главному меню…

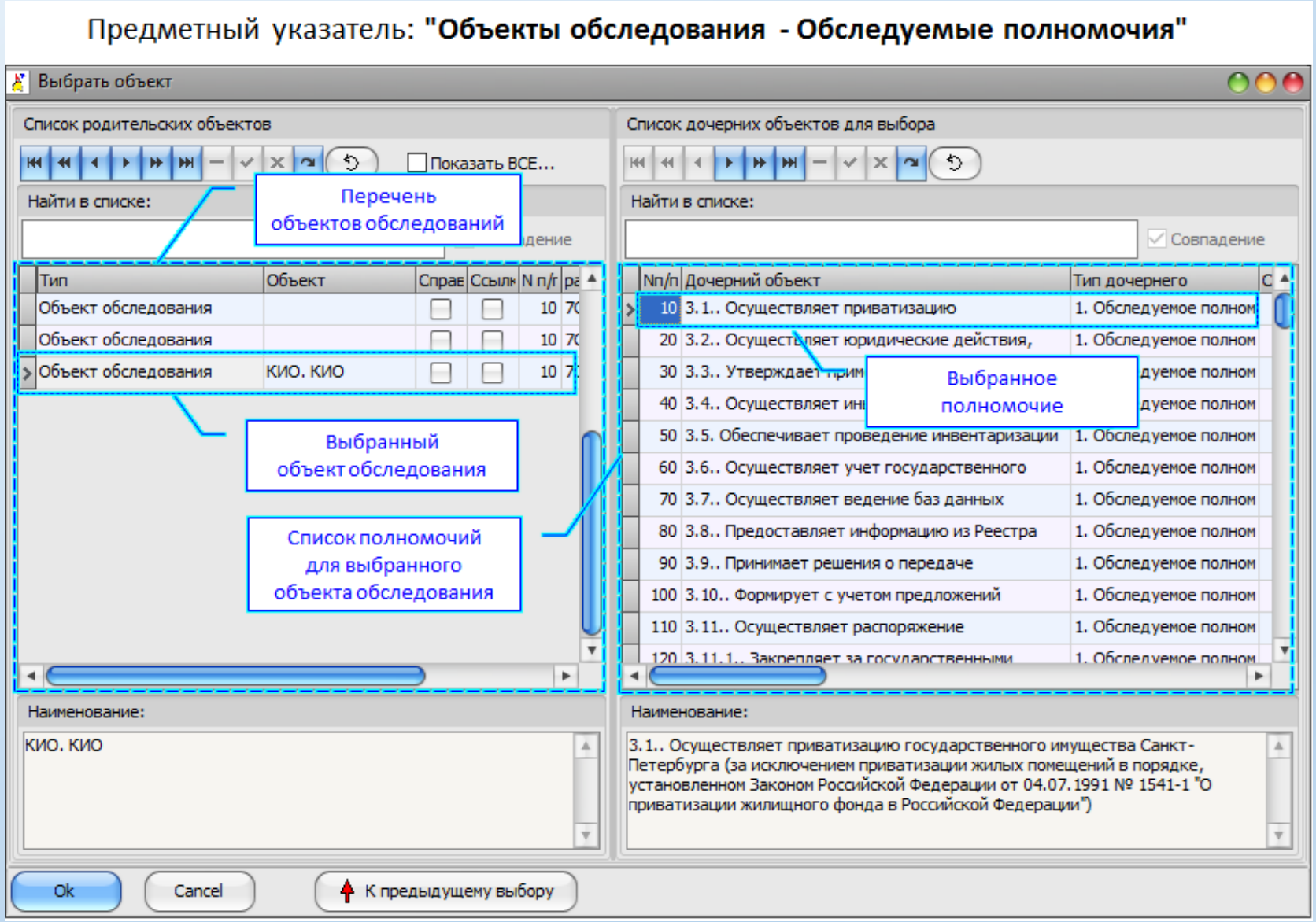
# Предметные указатели. Практическая реализация. **Форма выбора объекта**

У нас это реализовано так (см. рисунок).

**Важно!**

При всей простоте данной GUI-формы есть некоторая комплексная проблема, связанная с общими временнЫми затратами конечного пользователя.
А именно, поскольку эта форма вызывается под управлением соответствующего скрипта (см. здесь) N-е количество раз, то каждый раз принуждать пользователя «начинать поиск и выбор с начала» – не самая лучшая затея.
А значит, нужно радикально изменить структуру и состав данного скрипта (в настоящий момент еще не понятно, как именно).
А для этого, опять же, надо время, которого физически нет.

Т.е., в настоящий момент использование данного механизма вполне возможно, но он крайне не оптимизирован.




К главному меню...

# Интерпретаторы скриптов

**Интерпретатор скриптов** предназначен для реализации алгоритмов пользователя, связанных с обработкой информации, хранящейся в ИС.

При использовании адаптивной ИС предполагается, что ИС может обслуживать любую (заранее не определенную) предметную область.
Т.е., адаптивная ИС должна предоставлять гибкий инструментарий для:

**настройки** на соответствующую предметную область (см. предыдущие разделы);
**обработки** соответствующих данных в контексте соответствующих задач.

Применительно к нашей ИС, в качестве инструментария для **обработки** данных предназначены интерпретаторы скриптов:

уровень клиентского приложения (далее FS-интерпретатор) - создан на основе пакета FastScript for Delphi и функционирует только в среде ОС Windows;
уровень PostgreSQL-сервера (далее PG-интерпретатор) реализован на уровне plsql (и поэтому - межплатформенный).

**Важно!**

1. FS-интерпретатор – отработан достаточно НеПлохо для того, чтобы создавать скрипты, позволяющие генерировать отчеты (в частности, паспорта полномочий) в формате MS WORD с использованием механизма OLE Automation (для ОС Windows).
2. Именно потому, что существует требование: формировать сложно-структурированные отчеты (в формате MS WORD) с учетом сложного форматирования, то использование механизма OLE Automation представляется единственно реальным в предлагаемых обстоятельствах. Отсюда и зависимость от ОС Windows.
3. PG-интерпретатор – состояние его разработки носит **зачаточный характер** (т.е., почти на уровне «мысленного эксперимента»).
   Предназначен только для вычислительных задач на уровне PG-сервера.
   Предусматривалась возможность информационного обмена с клиентским приложением (платформа не имеет значения) с использованием механизма «встроенных» временных переменных.

К главному меню…

# Интерпретатор скриптов. **Уровень «клиента»**

**FS-интерпретатор** - создан на основе FastScript для Delphi и функционирует только в среде ОС Windows.
Ориентирован на конечного пользователя (уровень – системный аналитик).

Основные характеристики:

- объектно-ориентированный;
- при написании кода скрипта позволяет использовать как латиницу, так и кириллицу (русский язык);
- поддерживает синтаксис:
  - Pascal – тестировалось на нем (полноценный ObjectPascal);
- может поддерживать синтаксис (но **не** проверено):
  - C++;
  - JScript;
  - Basic;
- имеет встроенные:
  - редактор кода;
  - отладчик;
- содержит (в том числе) набор функций, ориентированных на объектную модель предметной области;
- поддерживает (в том числе) механизм OLE Automation;
- скрипты хранятся в ИБ (информационной базе);
- **не** использует Microsoft Scripting Host.

**Важно!**

1. Разработчик FastScript декларирует межплатформенность его использования (см. здесь: https://www.fastreport.ru/ru/product/fast-script/ ).
2. Но наши программные модули (макеты) выполнены в виде EXE-файлов под ОС Windows. Отсюда и зависимость от ОС Windows.

К главному меню…

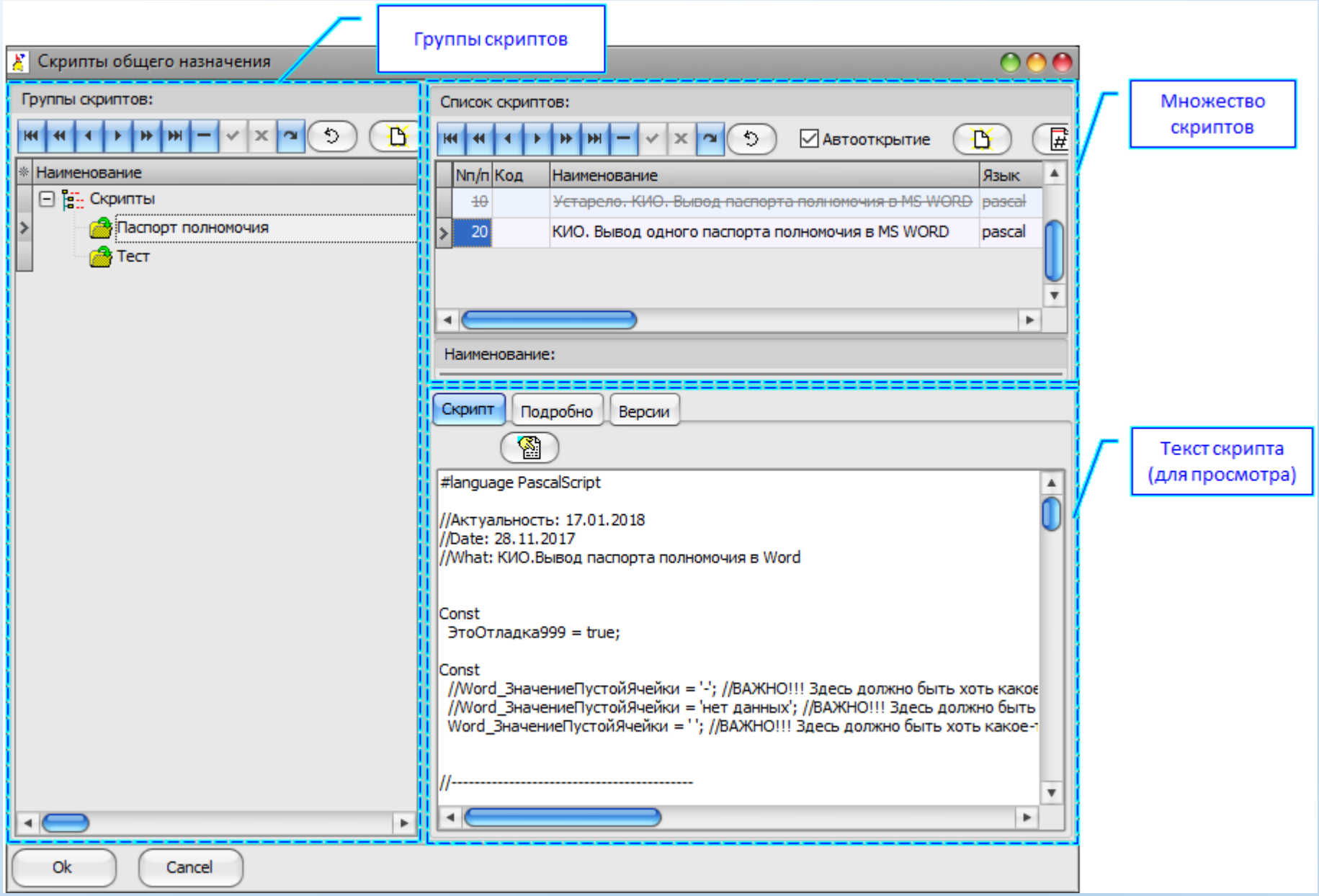
# Интерпретатор скриптов. Уровень «клиента». **Фрагмент скрипта в синтаксисе Pascal**

```
#language PascalScript

//Актуальность: 17.01.2018
//Назначение: КИО. Вывод паспорта полномочия в Word

Const
  Папка_ГдеWordШаблон = 'DOC_Templates\';
  Папка_КудаСохранитьПаспорт = 'DOC_Out\KIO\';
  Word_Шаблон_Имяфайла = 'KIO_Passport_template.doc';

function Паспорт_Полномочия_Вывод_в_Word:boolean;
Var
  Паспорт_Номер, Сервис_ПолноеИмя : string;
  Сервис_IdGuid:string;
  iSrv :integer;
begin
  if Начальная_Инициализация_и_Подготовка() then begin
     Экран_Курсор('crHourGlass');
     IdGuid_Объекта_Типа_ОбъединеннаяСтрока := trim(ТипОбъекта_Guid_по_Мнемокоду(Мнемокод_Объекта_Типа_ОбъединеннаяСтрока));
     vWord := Word_Открыть();
     vPasport := Word_ФайлОткрыть(vWord,Паспорт_Имяфайла);
     Паспорт_Номер := trim(Объект_Поле_Значение_КакСтрока_Получить(Паспорт_IdGuid, Паспорт_ItsDir, 'ИмяПолное'));
     Паспорт_Абзац_Добавить_Заголовок_Записать('ПАСПОРТ ПОЛНОМОЧИЯ '+Паспорт_Номер, -9, Word_Стиль_ЗаголовокПаспорта, '', '', true);
     Таблица_01_Вывод;
     СписокСервисовВПаспорте := TStringList.Create;
     СписокСценариевВСервисе := TStringList.Create;
     СписокДействийВСценарии := TStringList.Create;
     TRY
       СписокСервисов_Получить(СписокСервисовВПаспорте);
       Таблица_02_Вывод;
       Таблица_05_Вывод;
       iSrv:=-1;
       while iSrv<(СписокСервисовВПаспорте.Count-1) do
        begin
         iSrv:=iSrv+1;
         Сервис_IdGuid := trim(СписокСервисовВПаспорте.Names[iSrv]);
         Сервис_ПолноеИмя:= Сервис_ПолноеНаименование_Получить(Сервис_IdGuid);
         if iSrv>0 then Паспорт_Абзац_Добавить_Текст_Записать('','left', '', false);
         Паспорт_Абзац_Добавить_Заголовок_Записать(Сервис_ПолноеИмя, -9, Word_Стиль_ЗаголовокСервиса, '', '', false);
         Таблица_03_Вывод(Сервис_IdGuid);
         Таблица_15_Вывод(Сервис_IdGuid);
       end;
     FINALLY
       FreeAndNil(СписокСервисовВПаспорте);
       FreeAndNil(СписокСценариевВСервисе);
       FreeAndNil(СписокДействийВСценарии);
       Это_Конец_Формирования_Документа();
       Экран_Курсор('crDefault');
     END;
  end;
end;
```

К главному меню…

# Интерпретатор скриптов. Уровень «клиента»

## Практическая реализация «множества FS-скриптов»




К главному меню…

# Интерпретатор скриптов. Уровень «клиента»

## Практическая реализация интерпретатора скриптов (редактор кода)

КИО_Вывод_одного_паспорта_полномочия_в_MS_WORD

Файл скрипта  Скрипт  Панель сообщений  Help

Синтаксис: Pascal  Окно - поверх всех

Текст скрипта:

```
  end;

  СписокПунктов := TstringList.Create;
  ListCols := TstringList.Create;
  ListVals := TstringList.Create;
  ListRowsFull := TstringList.Create;
  TRY

    if Its07 then begin
       ListCols.Values['Структурное подразделение']:=FloatToStr(-1)+'|'+'умолчание';
       ListCols.Values['Должность']:=FloatToStr(-1)+'|'+'умолчание';
       ListCols.Values['Ф.И.О./Условный код участника']:=FloatToStr(-1)+'|'+'умолчание';
```

1 : 1

Панель сообщений:

Входные данные скрипта:

PrgDir=D:\pgFODB_exe_for_test\
TmpDir=TMP\

USER_ID_GUID=4CFD1FEBE41
USER_LOGIN=neon
SCRIPT_ID_GUID=B08AF4F2C
SCRIPT_MNEMO=КИО_Вывод_

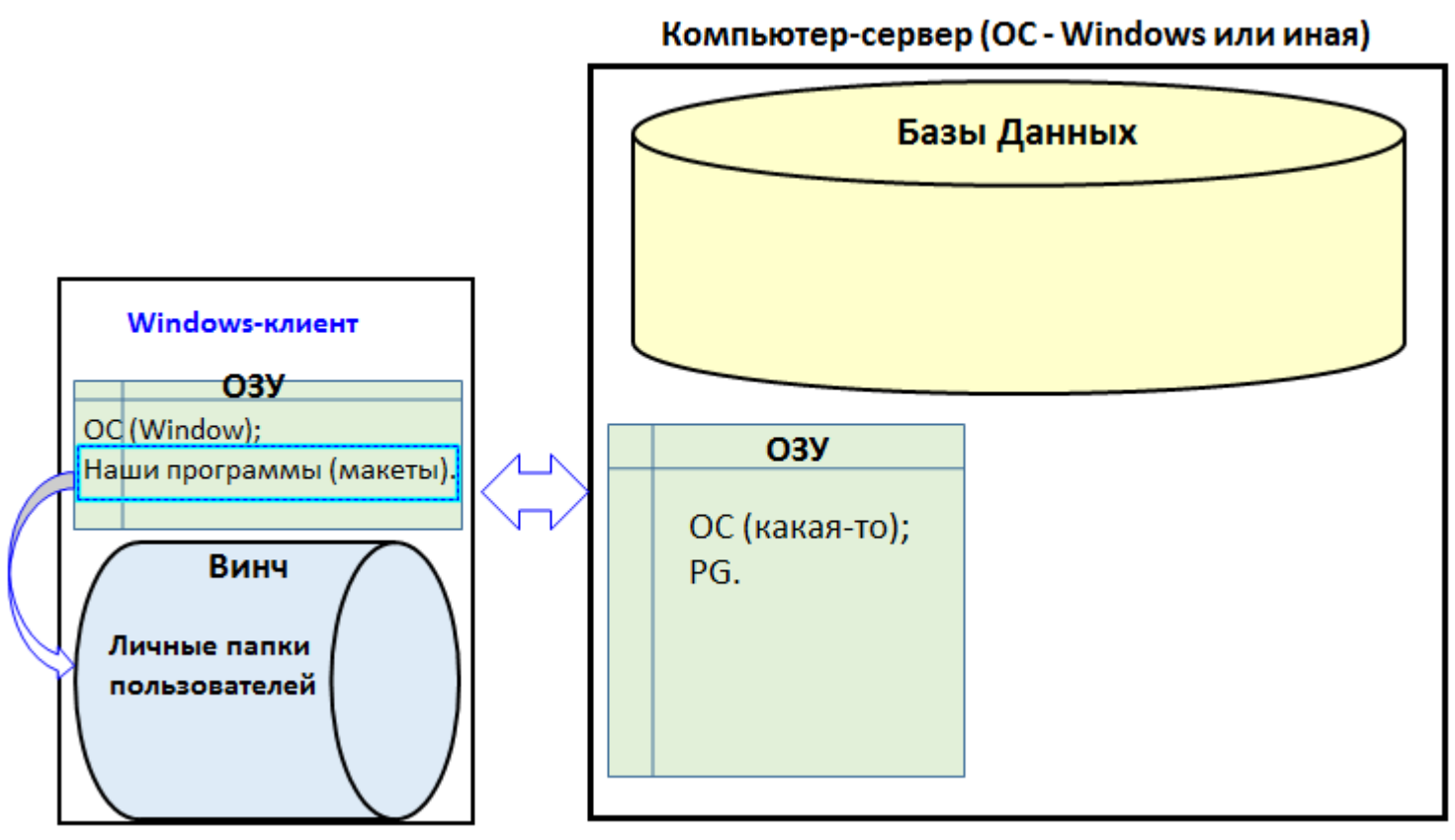
Ok  Cancel  Есть соединение с 195.144.238.224:kio_01

К главному меню…

# Интерпретатор скриптов. Уровень «клиента».

## Варианты решения проблемы зависимости от ОС Windows

На рисунке ниже представлена обобщенная схема, иллюстрирующая функционирование клиентского ПО в среде ОС Windows (windows-модули, EXE). Т.е., это то, как мы работает сейчас (с нашими программными макетами).




Т.е., в данном случае никаких проблем (связанных с «зависимостью») - нет.

К главному меню…

# Интерпретатор скриптов. Уровень «клиента».

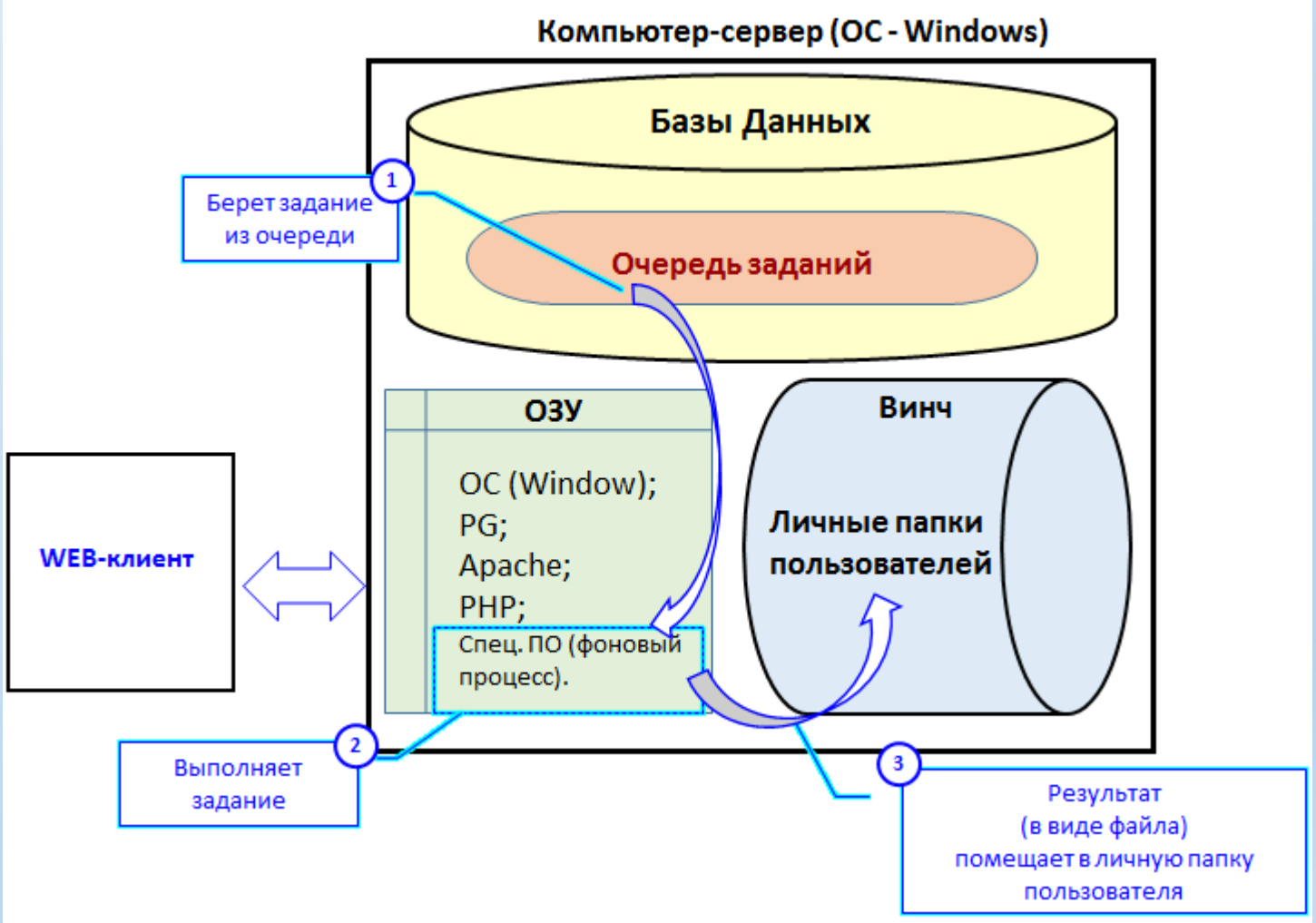
## Варианты решения проблемы зависимости от ОС Windows

На рисунке представлена обобщенная схема, иллюстрирующая функционирования клиентского ПО (WEB-браузер).
Сервер функционирует под управлением ОС Windows.

**Важно!**

То, что на «картинке» - это уровень «декларации намерений». Т.е., его еще надо методически, программно и технически прорабатывать.

Но каких-то существенных сложностей не просматривается (хотя сам объем работ будет не мал).

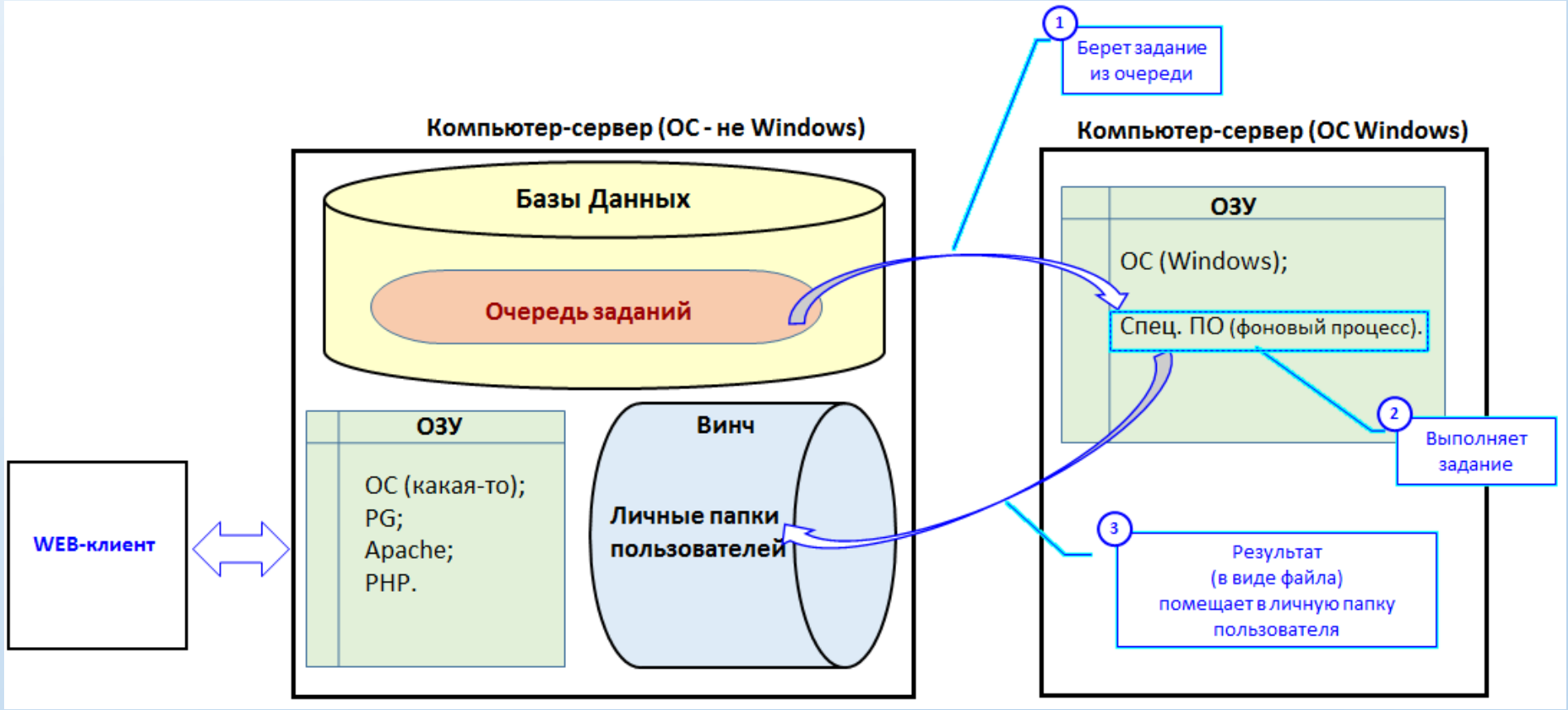



К главному меню…

# Интерпретатор скриптов. Уровень «клиента».

## Варианты решения проблемы зависимости от ОС Windows

На рисунке представлена обобщенная схема, иллюстрирующая функционирования клиентского ПО (WEB-браузер).
Основной компьютер (сервер), где развернут PostgreSQL, функционирует под управлением «какой-то ОС».
Дополнительный компьютер функционирует под управлением ОС Windows.




**Важно!**

То, что на «картинке» - это уровень «декларации намерений».
Т.е., его еще надо методически, программно и технически прорабатывать.
Но каких-то существенных сложностей не просматривается (хотя сам объем работ будет не мал).

К главному меню…

# Интерпретатор скриптов. **Уровень PostgreSQL-сервера**

Назначение **PG-интерпретатора** – снять зависимость (насколько это возможно) клиентского ПО от Windows при обработке информации в ИБ (информационной базе).

Что понимается под термином «обработка информации в ИБ»:

1. Чтение значений каких-то численных свойств каких-то объектов, а также (факультативно), чтение каких-то входных параметров.
2. Их обработка (вычислительные операции с использованием формул, циклов и условных операторов).
3. Запись (факультативно) вычисленных значений в какие-то свойства каких-то объектов.
4. Запись (факультативно) вычисленных значений в какие-то выходные параметры.
5. Возможность для клиентского приложения передать в скрипт входные параметры и прочитать от скрипта выходные параметры.

Предусматривался следующий алгоритм работы с PG-скриптами на клиентском ПО (см. рисунок на слайде далее).

Примеры PG-скриптов см. далее здесь и здесь.

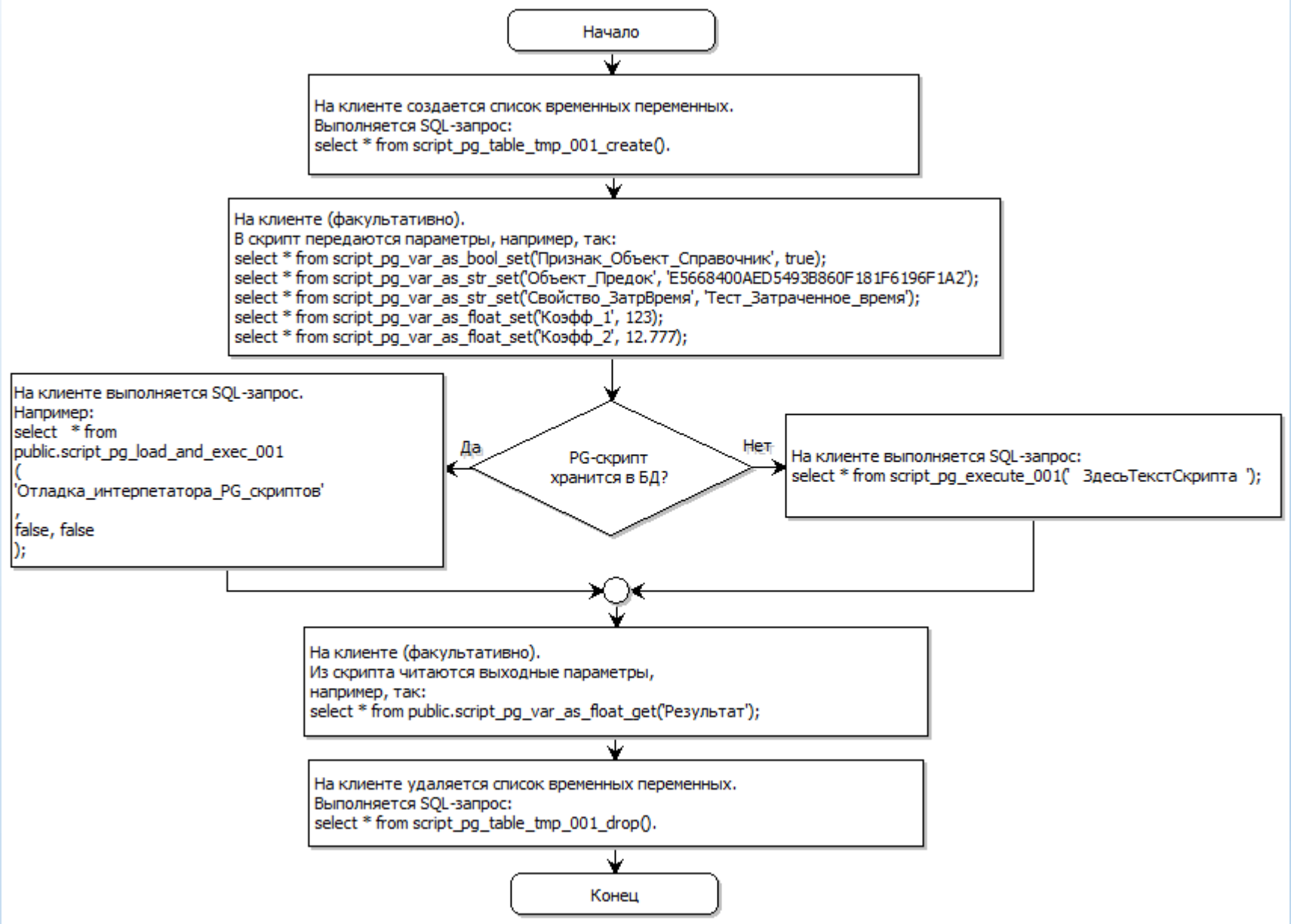
**Важно!**

Состояние разработки PG-интерпретатора носит зачаточный характер (т.е., почти на уровне «мысленного эксперимента»).

К главному меню…

# Интерпретатор скриптов. **Уровень PostgreSQL-сервера**

Алгоритм работы с PG-скриптами на клиентском ПО (см. рисунок).




К главному меню…

# Интерпретатор скриптов. Уровень PostgreSQL-сервера. **Пример скрипта**

**Текст скрипта**

```
m:=0;
s:=0;
z:=0;
j:=0;
WHILE z<10 do
  z := z + 1;
  j:=0;
  WHILE j<10 do
    j := j + 1;
    s := s+1;
    if s <= 3 then
       m:=m+1;
    else
       m := m+10;
    end_if;
    a:=333.4;
    b := 444;
    d:=55;
    x:=(a + b) / (a - b) * d + 17.5;
    IF X>99 THEN
       b := x-5;
       IF b>3 THEN
          d:=34;
       ELSE
          d:= 72;
       END_IF;
    END_IF;
  END_WHILE;
END_WHILE;
```

К главному меню…

# Интерпретатор скриптов. Уровень PostgreSQL-сервера. **Пример скрипта**

**Входные параметры:**

```
select * from script_pg_var_as_bool_set('Признак_Объект_Справочник', true);
select * from script_pg_var_as_str_set('Объект_Предок', 'E5668400AED5493B860F181F6196F1A2');
select * from script_pg_var_as_str_set('Типы_Объектов_Потомков', 'Объект_1_для_отладки_script_pg_execute');
select * from script_pg_var_as_str_set('Свойство_ЗатрВремя', 'Тест_Затраченное_время');
select * from script_pg_var_as_str_set('Свойство_ВыпРабота', 'Тест_Выполненная_работа');
```

**Текст скрипта**

```
Результат := "???";
Какойто_Коэфф := 1.2;
Затраты_Времени:=ОбъектыСвойствоЗначениеСуммаПолучить(Признак_Объект_Справочник, Объект_Предок, Типы_Объектов_Потомков, Свойство_ЗатрВремя);
Выполн_Работа:=ОбъектыСвойствоЗначениеСуммаПолучить(Признак_Объект_Справочник, Объект_Предок, Типы_Объектов_Потомков, Свойство_ВыпРабота);
Работа_В_Среднем := "Хрень какая-то";
IF Затраты_Времени > 0 THEN
   Работа_В_Среднем := (Выполн_Работа/Затраты_Времени) * Какойто_Коэфф;
   IF Работа_В_Среднем > 5 THEN
      Результат := "Многовато";
   ELSE
      IF Работа_В_Среднем <= 1 THEN
         Результат := "Маловато";
      ELSE
         Результат := "Норм";
      END_IF;
   END_IF;
   * ------------------------------------
   * Это комментарий
   * Иллюстрация цикла
   Сумма_В_среднем := Работа_В_Среднем;
   i:=0;
   WHILE i<20 DO
     i:=i+1;
     Сумма_В_среднем := Сумма_В_среднем + i;
   END_WHILE;
   * ------------------------------------
END_IF;
```

**Выходные параметры:**

```
-- Прочитать значения всех переменных скрипта
select * from scr_pg_001_tt999 order by var_name;

-- Прочитать значения конкретных переменных скрипта
select * from script_pg_var_as_float_get('Работа_В_Среднем');
select * from script_pg_var_as_float_get('Результат');
```

К главному меню…

# Информационные объекты и их свойства

**Объект (информационный)** – это совокупность адаптивной и логически целостной информационной структуры, отражающей в информационном пространстве Системы реальный объект из предметной области, и адаптивных алгоритмов, определяющих специфику его информационной поддержки (под адаптацией понимается возможность оперативных изменений информационных структур и алгоритмов без перепрограммирования Системы и реструктуризации таблиц базы данных).
Объект обязательно имеет родителя (любой другой, уже существующий в Системе, объект) и тип (из множества типов, "известных" Системе), а также имеет набор системно-предопределенных свойств и может обладать свойствами из множества свойств, определяемых пользователем. Исключение составляет корневой объект (в дереве объектов), который не имеет родителя (корневой объект может быть только один в дереве).

**Абстрактный объект** – это не существующий в предметной области объект, которым оперирует пользователь для целей группировки, обобщения и классификации информации в Системе.

*Реальные объекты предметной области могут проявлять (для наблюдателя) свои свойства в контексте других объектов, а также, в зависимости от условий исследования (наблюдения), воздействия и представления - по разному.*
*При этом, это один и тот же объект.*
*Т.е., любой объект (являясь целостным, сложным и непрерывным в "пространстве-времени") может быть по разному представлен и "информационно усечен", в зависимости от задач.*

**Ссылочный объект** (ссылка) - это объект, ссылающийся на любой другой объект (исключая самого себя) и наследующий его некоторые свойства (тип, наименование и т.д.) из перечня системно предопределенных свойств. Ссылочный объект может обладать своим набором свойств, определяемых пользователем.

Любой объект по отношению к своим свойствам является **владельцем**.
**Важно!** Не следует путать свойства, входящие «в множество свойств» и свойства, владельцем которых является объект.

Объекты условно разделены на **Объекты** и **Объекты-Справочники**.
Объекты (и их свойства) размещены в схеме **ws** (ws.obj_tree – объекты; ws.obj_props – свойства).
Объекты-справочники (и их свойства) - в схеме **dir** (dir.dir_tree – объекты-справочники; dir.dir_props – свойства).
Везде, где нет существенного различия между терминами "Объект" и "Объект-справочник", в тексте данного документа используется термин "Объект".

К главному меню…

# Информационные объекты и их свойства

Совокупность объектов в информационной базе представляет собой **иерархическое дерево объектов**.

Любой объект может быть «родителем/предком» для концептуально неограниченного количества «потомков» любого типа.
Т.е., сложность и глубина иерархического дерева концептуально не ограничены.

При этом, отношения объектов "Родитель" - "Дочерние", в контексте дерева объектов, следует интерпретировать следующим образом:

"Дочерний" объект **актуален** в контексте "Родительского" объекта.

Под **актуальностью** понимается конкретная реализация свойств (и/или их значений) "дочернего" объекта, проявляющаяся в контексте "родительского" объекта.

**Предок** - объект, в информационную структуру которого включены другие объекты, уровень иерархии которых больше, чем уровень иерархи предка.
Термин "Предок" актуален только в контексте "Потомков".

**Родитель** (родительский объект) - объект, в информационную структуру которого включены другие объекты, уровень иерархии которых на единицу больше, чем уровень иерархии родительского объекта. Термин "Родитель" актуален только в контексте "Дочерних объектов".

**Дочерний объект** - объект(ы), имеющий(е) родителя (уровень иерархии дочерних объектов всегда больше единицы).

**Потомок** - объект(ы), имеющий(е) предка (уровень иерархии потомков всегда больше единицы). Дочерние объект - частный случай «потомков».

Любой дочерний объект (или потомок) может быть родителем или предком для других объектов любого типа.

**Важно!**

Предусмотрен механизм «привязки» по определенным правилам «сторонних» таблиц БД (не входящих в состав ИС) к объектам в дереве объектов (на уровне триггеров таблиц БД: dir.dir_tree_ext и ws.obj_tree_ext).
Цель – предусмотреть возможность расширения ИС (при необходимости) в сторону «жестких» решений конкретных («узких»), специфичных задач.

К главному меню…

# Информационные объекты и их свойства

Если не вдаваться в технические подробности, то по рисункам на слайдах здесь и здесь можно составить вполне адекватное представление, как выглядит дерево объектов и перечень свойств, владельцем которых является какой-либо объект.

К главному меню…

# GUI

**Аксиома.**
В общем случае, «узко-заточенная» ИС (разработанная под конкретную прикладную задачу) по **`usability`** превзойдет адаптивную (или претендующую на ее роль) ИС, используемую для этой же задачи.

При создании адаптивной ИС одной из существенно знАчимых проблем является создание адаптивного (и объектно-ориентированного на предметную область) GUI приемлемого «качества».

*Примечание – вообще, по сегодняшним временам, термин «адаптивный и объектно-ориентированный на предметную область GUI» звучит более, чем странно. Начни с кем-то обсуждать – в лучшем случае, посмотрят на тебя с «пониманием» (типа – человек с проблемами в голове), а в худшем – покрутят пальцем у виска и скажут – «не отнимай наше время, а лучше займись делом».*

В текущий момент времени в нашей ИС существуют (применительно к объектам и их свойствам) следующие варианты GUI:

1. Дерево объектов;
2. Рабочий список пользователя;
3. Настраиваемый оконный GUI;
4. Универсальная, управляемая GUI-форма;
5. Жесткий вариант реализации действующего макета приложения, предназначенного для проведения самообследования.

**Важно!**

1. Каждый из этих вариантов имеет как преимущества, так и недостатки. А также, различную степень проработки, тестирования и готовности (если можно так сказать).
2. Все, выше перечисленные варианты (кроме варианта 3), по любому используют вариант «Дерево объектов», но с той разницей, что объем загружаемых данных может управляемо регулироваться.
3. Поскольку, в конечном итоге, пользователь работает с «Деревом объектов», то (не зависимо от предметной области), GUI всегда один и тот же. Т.е., нет необходимости его «изучения» при переходе от одной предметной области к другой даже в оперативном режиме. Что является несомненным плюсом.

К главному меню…

# GUI. **Дерево объектов**

Данный вариант (в нашем случае) предполагает сразу полную загрузку всего дерева объектов со всеми вытекающими.

Основной **плюс** в том, что этот вариант является наиболее универсальным для использования и простым для реализации.

Главный (и решающий) **минус** в том, что при значительном объеме данных этот вариант будет просто не применим (и особенно для WEB-версии Системы).

**Важно**

1. Существуют известные технические методы, которые позволяют в какой-то мере купировать этот «минус». Например, можно дерево загружать частично (допустим, сначала только 1-й уровень иерархии и, далее, по мере того, как конечный пользователь будет выбирать очередной узел – перестраивать дерево таким образом, чтобы выбранный узел отображался на верхнем уровне и загружались его дочерние узлы, и т.д.). Есть и другие методы.
2. Но это только частичное решение проблемы.

Но тем не менее, у нас этот вариант есть (исторически – он был самым первым и самым востребованным).
И, пожалуй, он (на текущий момент) самый проверенный (но в общем случае – это тупиковый вариант).

К главному меню…

# GUI. Дерево **объектов-справочников**. Практическая реализация

Дерево объектов

Объект: Справочники

2637B8CD7A6F40CDB6

Дерево объектов

Свойства выбранного объекта

Объединенная строка (Word)

| Тип объекта | Наименование |
|---|---|
| Корневой раздел "дерева" | Справочники |
| Справочник организаций | |
| Организация | АПЕТР |
| Список структурных подразделений | |
| Отдел | АПЕТР. ЖО |
| Списочный состав | |
| Сотрудник | АПЕТР. ЖО. Воробьева А.В.. На |
| Сотрудник | АПЕТР. ЖО. Исакова Е.В.. Главн |
| Сотрудник | АПЕТР. ЖО. Габриелян К.В.. Гла |
| Отдел | |
| Отдел | |
| Список ИС | |
| Информационная система | . Тип: Прочие ИС |
| Список информационных пакетов | |
| Информационный пакет | Информация о льготах заявител |
| Информационный пакет | Информация о предоставлении у |
| Список исполняемых полномочий | |
| Исполняемое полномочие | Обеспечивать рассмотрение обр |
| ГУ Государственная услуга | Обеспечение рассмотрения обра |
| ГФ Государственная функция | Обеспечивать рассмотрение обр |
| Иное | - |
| Исполняемое полномочие | Осуществляет приватизацию го |
| Исполняемое полномочие | Организовывать проведение ме |

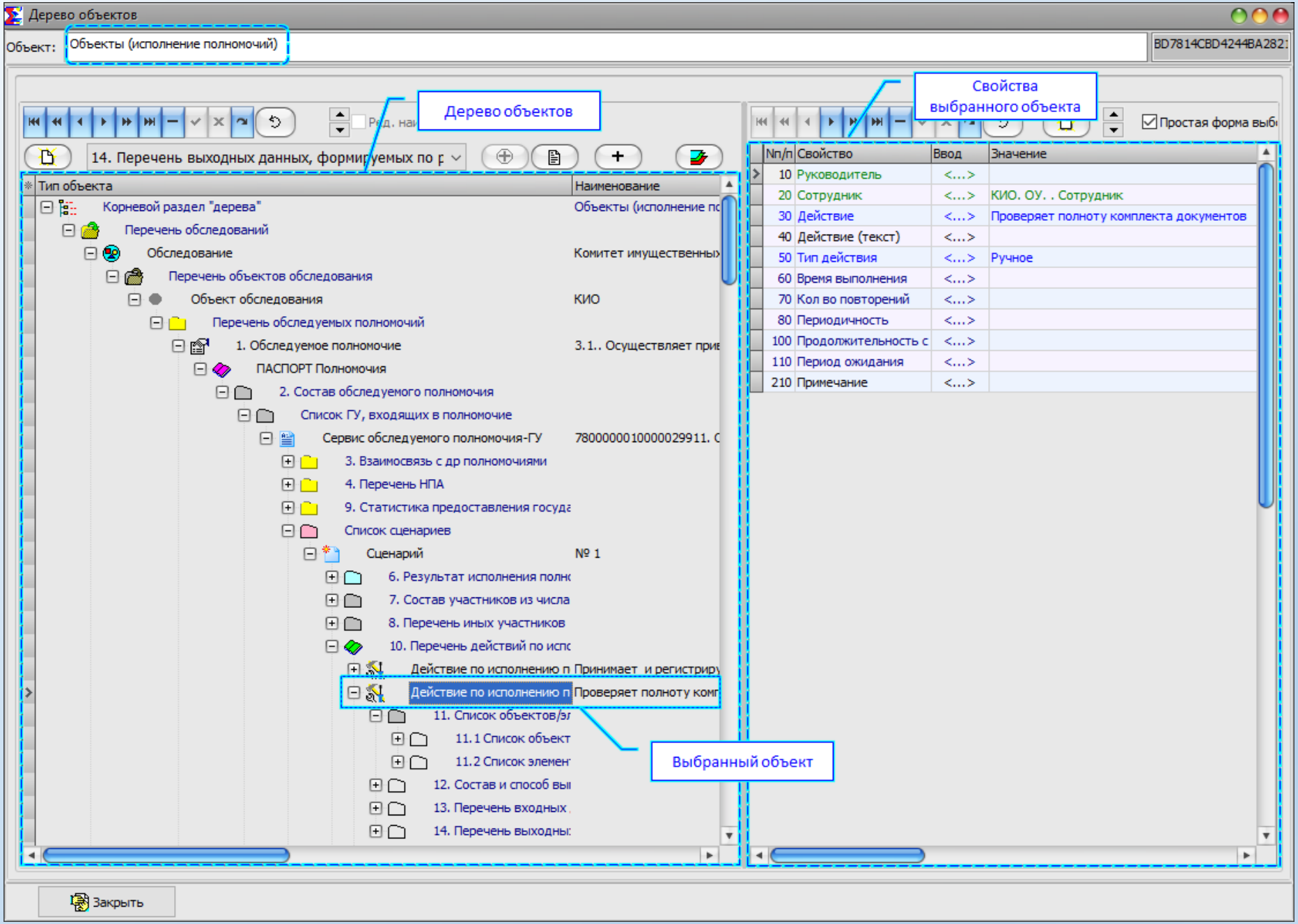
Выбранный объект

| Nп/п | Свойство | Ввод | Значение |
|---|---|---|---|
| 10 | Должность | <...> | Начальник отдела |
| 20 | Ф.И.О. | <...> | Воробьева А.В. |
| 30 | Ф.И.О. (сокращенно) | <...> | Воробьева А.В. |
| 40 | Логин | <...> | |

Закрыть

К главному меню...

# GUI. Дерево **объектов**. Практическая реализация




К главному меню…

# GUI. **Рабочий список пользователя**

Основное назначение: содержит перечень персональных заданий, связанных с обработкой данных в контексте конкретных объектов.

Под заданием, в данном случае, понимается конкретный объект, определяющий контекст (отправную точку) работы с деревом объектов.

Предполагались следующие варианты использования:

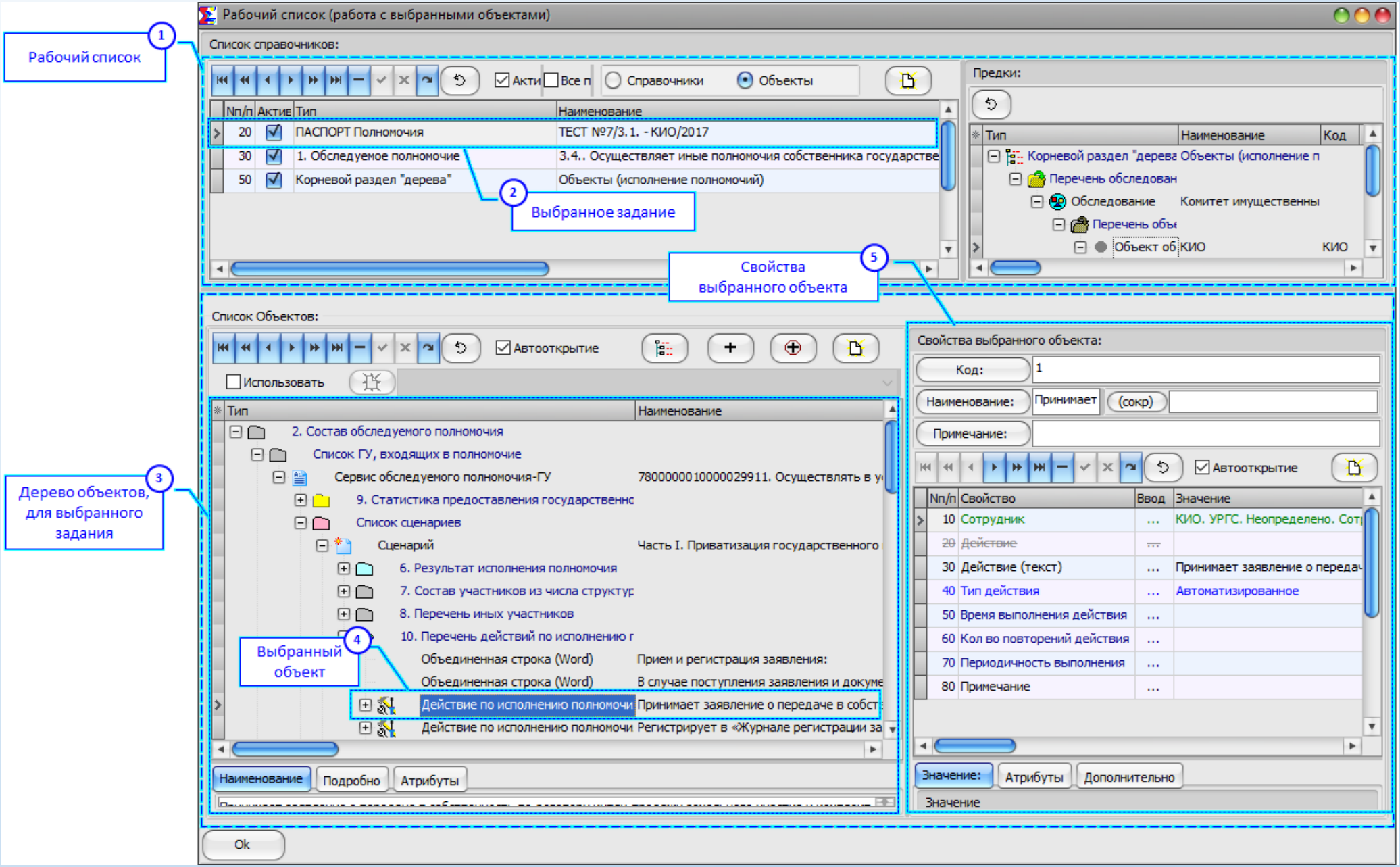
1. Пользователь-исполнитель сам отбирает себе соответствующие объекты в свой рабочий список (т.е., самостоятельно формирует свой рабочий список) и, далее, в оперативном режиме работает с отобранными объектами.
2. «Распределитель работ» формирует рабочие списки персонально для каждого пользователя-исполнителя.
3. Оба выше перечисленных варианта могут использоваться совместно.

**Важно!**

1. Не следует путать «Рабочий список» и «Технологическая цепочка распределенного ввода поступающей информации». Назначение у них схожее, но возможности, методы использования, сложность реализации и объем работ по реализации – совершенно разные. «Технологическая цепочка» знАчимо превосходит «Рабочий список» по всем перечисленным показателям (включая и затраты).
2. Вариант, когда Пользователь-исполнитель сам формирует свой рабочий список, в настоящий момент отработан в достаточной (для практического использования) мере.
3. Вариант с «Распределителем работ» не реализован, поскольку в практической работе (когда с ИС работали только два человека) этого не требовалось. Но на его реализацию не требуется много времени.

К главному меню…

# GUI. Рабочий список пользователя. **Практическая реализация**




К главному меню…

# GUI. **Настраиваемый оконный GUI**

Основное назначение: буквально реализация желания Метельковой «Работать (если это возможно) с отдельными, простыми таблицами, а не в дереве».
В принципе, такой вариант GUI показался интересным и полезным «вообще», поэтому это было реализовано (в том числе, чтобы пощупать все плюсы/минусы на практике).

## Основные идеи реализации:

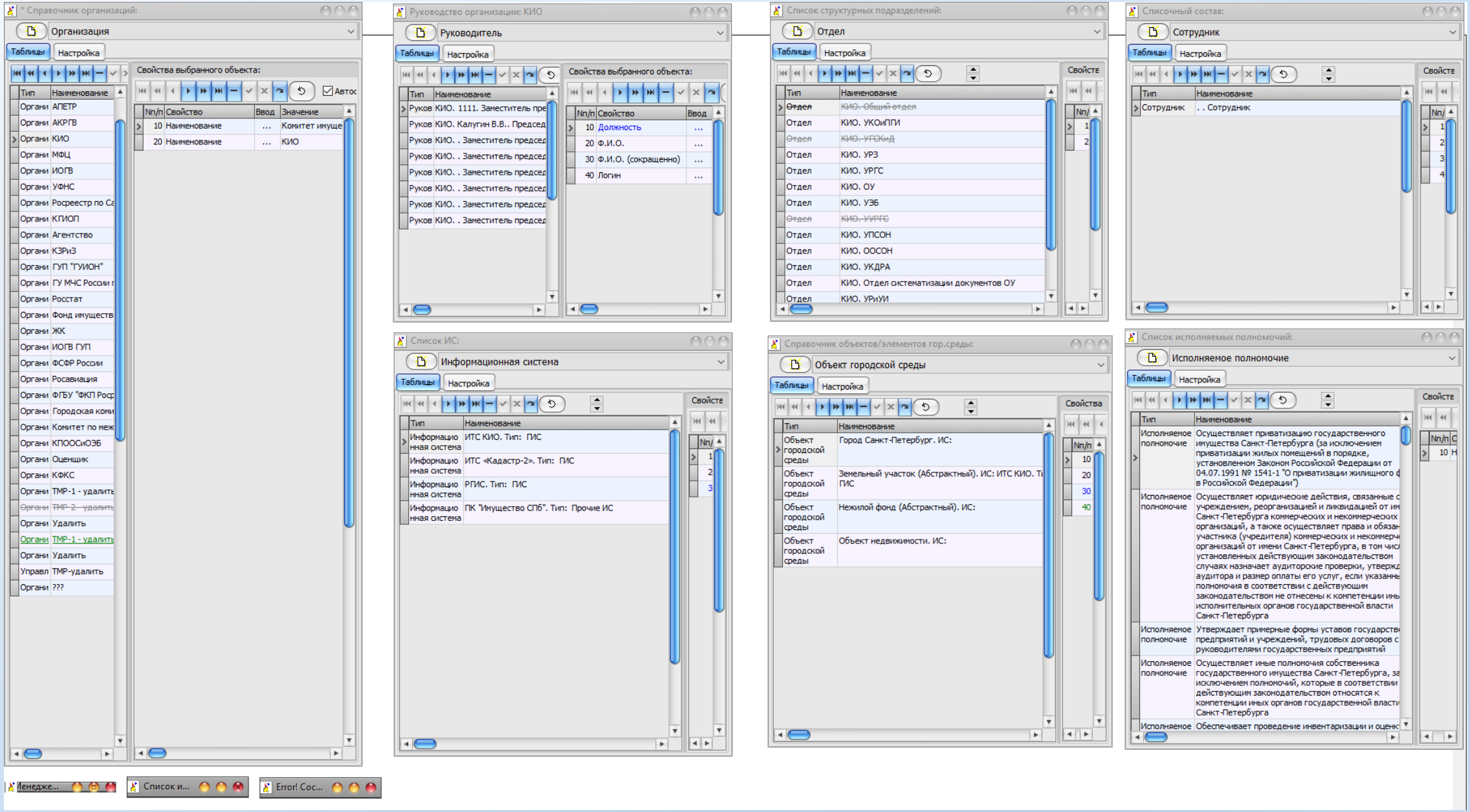
1. Использование оконного GUI должно быть факультативным.
2. Формирование «оконного GUI» должно производиться Аналитиком интерактивно и по определенным правилам.
3. Должна быть предоставлена (для Аналитика) возможность формирования любого (концептуально не ограниченного) количества вариантов «оконного GUI», содержащего любое (концептуально не ограниченное) количество окон, для любого типа объектов (из предметной области).
4. Формирование «оконного GUI» концептуально не должно зависеть от предметной области.
5. При использовании какого-либо варианта «оконного GUI» конкретным пользователем – должны сохраняться основные параметры каждого окна таким образом, чтобы при повторном открытии данного варианта «оконный GUI» «восстанавливался».

**Важно!**

1. Реализовать-то удалось, но никто его не тестировал, в общем-то.
   И, как следствие, никто не может однозначно сказать: в какой степени готовности (для практического использования) это находится.
   И вообще – насколько и в чем он «лучше» (или «хуже»), чем иные варианты GUI.
2. В целом, такой вариант GUI представляется полезным. Но для WEB-версии, боюсь, он **не** достижим (если оценивать реально).

К главному меню…

# GUI. Настраиваемый оконный GUI. **Практическая реализация**




К главному меню…

# GUI. **Универсальная, управляемая GUI-форма**

**Унифицированная, управляемая форма** (ууФ) – приложение, предназначенное для автоматического формирования унифицированного GUI-интерфейса на основе настроек, интерактивно созданных конечным пользователем Системы.
Пользователь, формируя какой-либо вариант соответствующих настроек, тем самым управляет «поведением» унифицированного GUI-интерфейса.
Количество вариантов настроек концептуально не ограничено.
Цель: Поступающая для ввода данных информация может быть существенно разнородной, что потребует ее разделения на отдельные части, как по типовому признаку, так и по последовательности обработки.
Это неизбежно приведет к необходимости формирования некоторой технологической цепочки обработки данных, включающей (в том числе) создание соответствующего количества специфичных рабочих мест.
Использование ууФ предоставляет возможность пользователю оперативно создавать такие рабочие места (формируя соответствующие настройки).

Идея заключается в следующем:

1. GUI – это предметная область, включающая определенные объекты и их свойства.
2. Эта предметная область (GUI) – ничем (концептуально) не отличается от иных предметных областей, которые обслуживаются или могут обслуживаться адаптивной ИС.
3. Следовательно, можно предположить, что в основу формирования адаптивного механизма для ууФ вполне может быть положена идея: провести ассоциативное связывание объектов и их свойств из предметной области GUI с объектами и их свойствами из обслуживаемой (адаптивной ИС) предметной области.
4. В этом случае, необходимо рассматривать приложение ууФ, как некий интерпретатор, исполняющий команды аналитика, производящего соответствующие настройки.
5. Очевидное требование: информ. модель предметной области GUI должна быть достаточно полно проработана и исчерпывающе четко определена до того, как будет начата разработка приложения ууФ и, в дальнейшем, любые существенные изменения в информ. модели GUI должны быть предварительно согласованы с разработчиком ууФ.

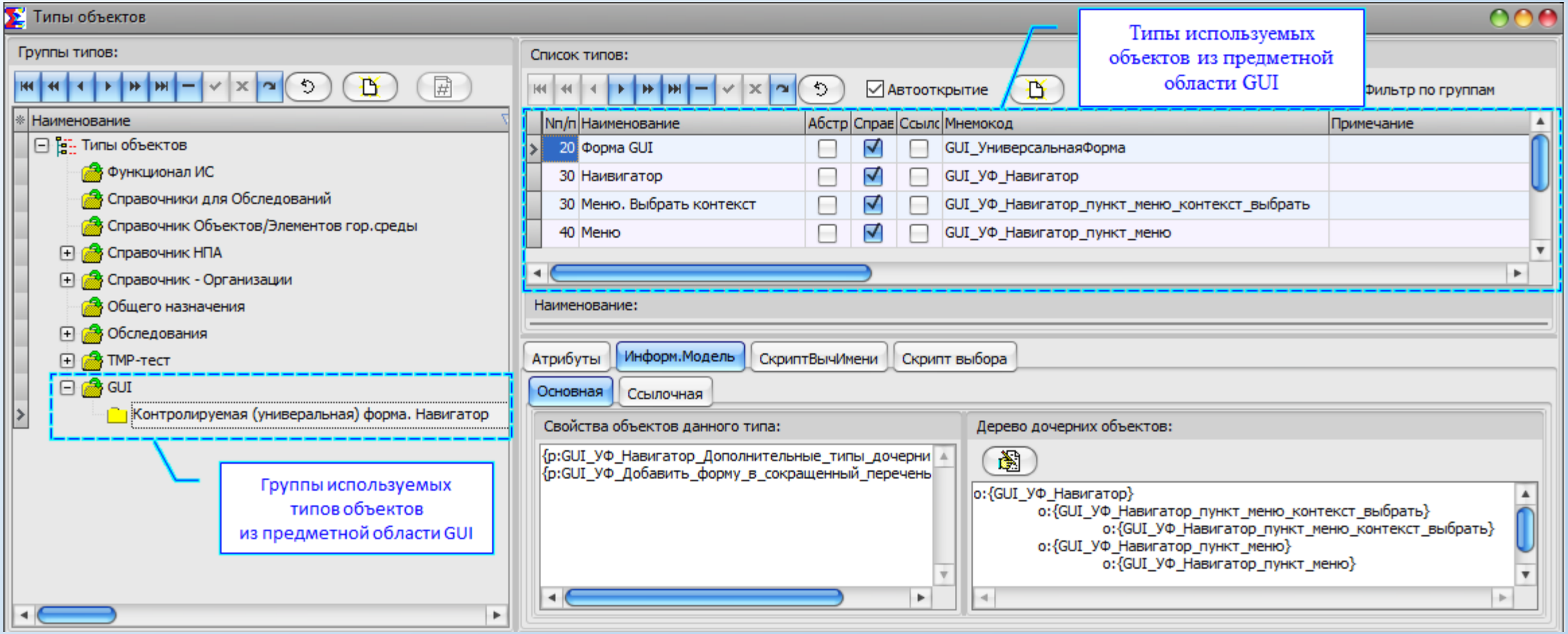
## Важно!

1. Идея, в общем-то, оказалась «живучей».
2. В настоящий момент есть рабочий макет (программный модуль) приложения ууФ (где сделана попытка как-то «пощупать» это на практике).
3. По результатам очень поверхностной работы с ним можно определенно утверждать, что в принципе техническая реализация ууФ (на основе выше сказанного) - вполне возможна и оправдана.
4. При этом, исчерпывающе полного и ясного понимания, что это такое (и как его реализовывать на практике) «ассоциативное связывание объектов и их свойств из предметной области GUI с объектами и их свойствами из обслуживаемой (адаптивной ИС) предметной области» лично у меня на сегодня **нет**.
5. Следствие:
   5.1 Если планировать выход на «Технологические цепочки», то есть безальтернативная необходимость предварительно иметь готовый и отлаженный механизм ууФ (или чего-то подобного) со всеми вытекающими (надо «копать» дальше).
   5.2 В настоящий момент **не** следует рассматривать действующий макет ууФ, как готовый для практического использования.

К главному меню…

# GUI. ууФ. Практическая реализация. **Настройка**

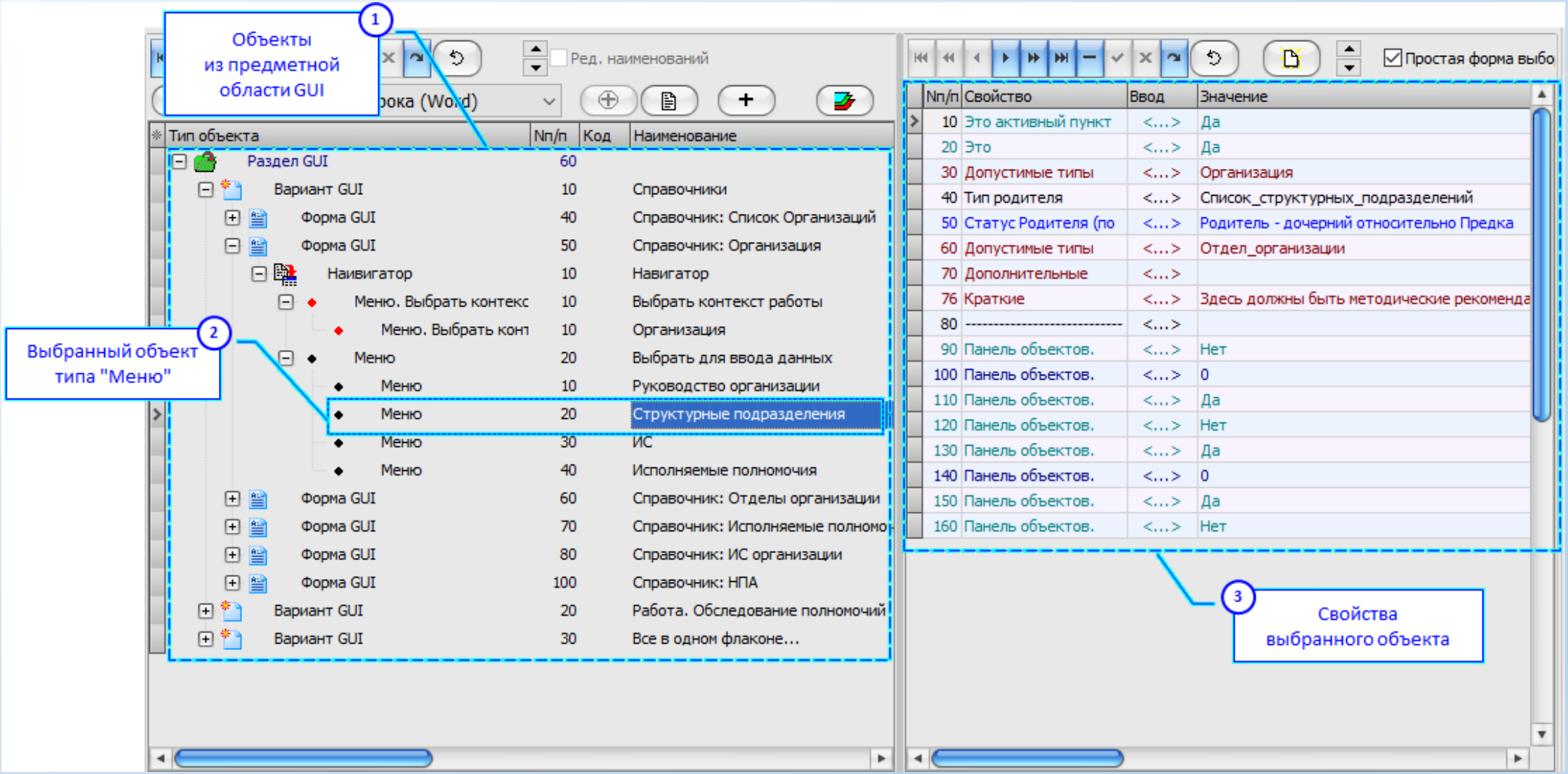
На рисунке ниже представлено подмножество используемых типов объектов из предметной области GUI.




К главному меню…

# GUI. ууФ. Практическая реализация. **Настройка**

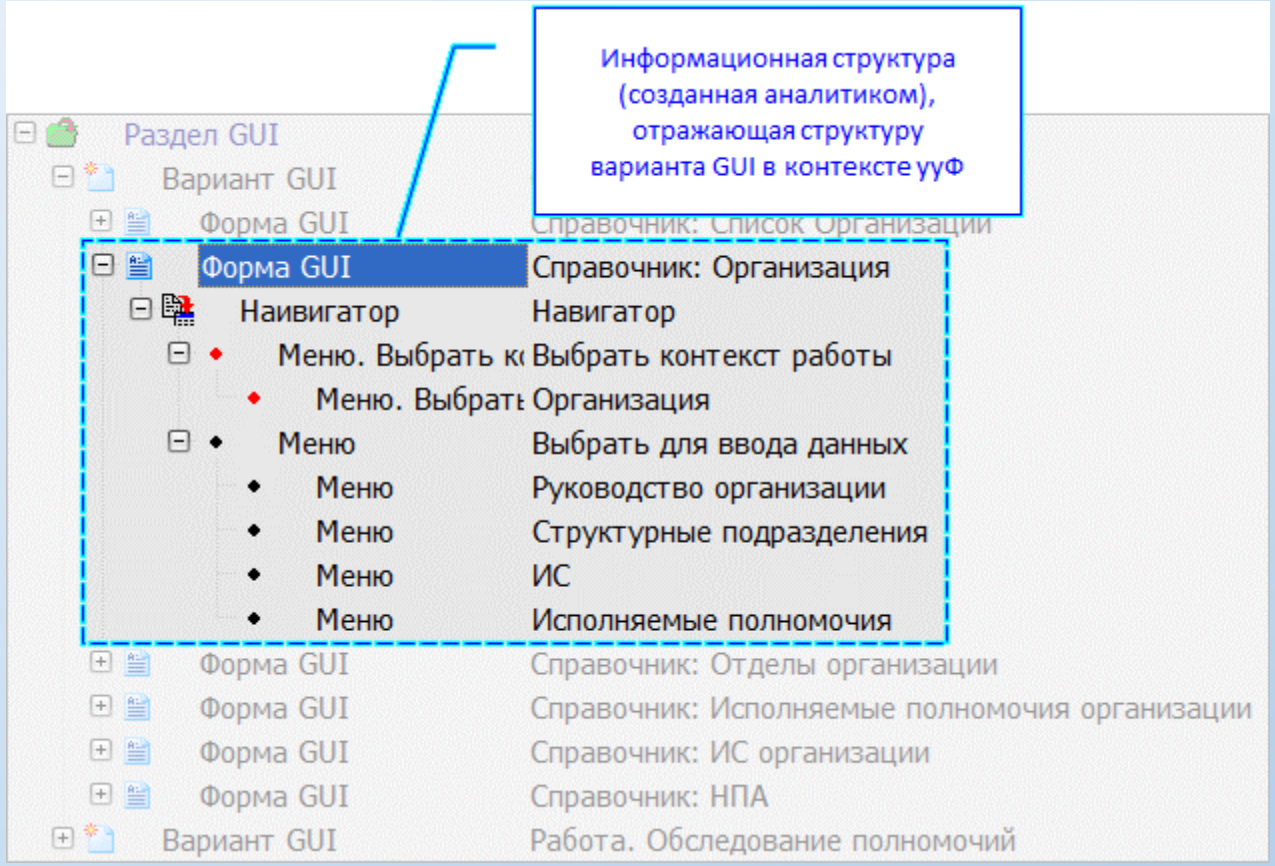
На рисунке ниже представлено дерево объектов из предметной области GUI.




К главному меню...

# GUI. ууФ. Практическая реализация. **Настройка**

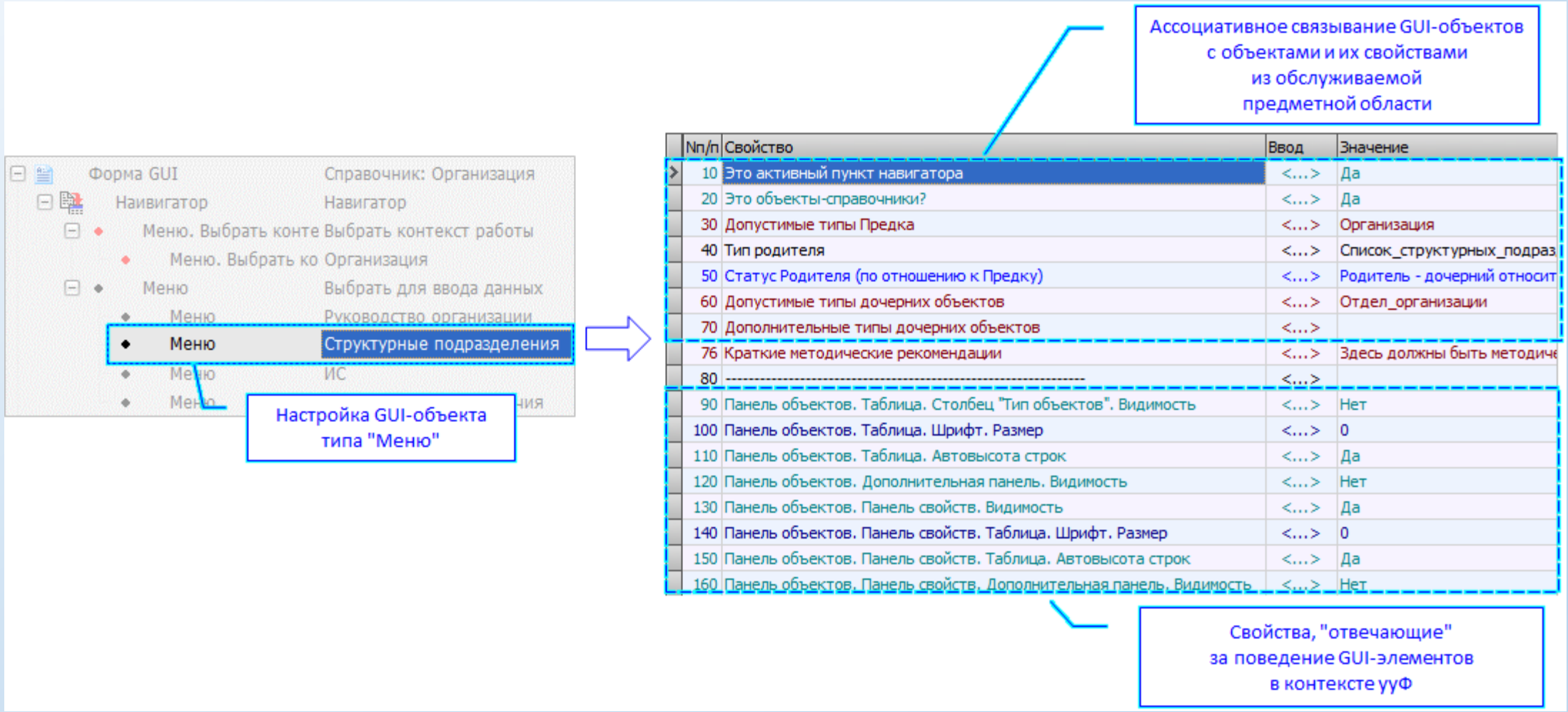
На рисунке ниже представлена информационная структура объекта типа «Форма GUI» (интерактивно созданная аналитиком), отражающая структуру данного варианта GUI в контексте ууФ.




К главному меню…

# GUI. ууФ. Практическая реализация. **Настройка**

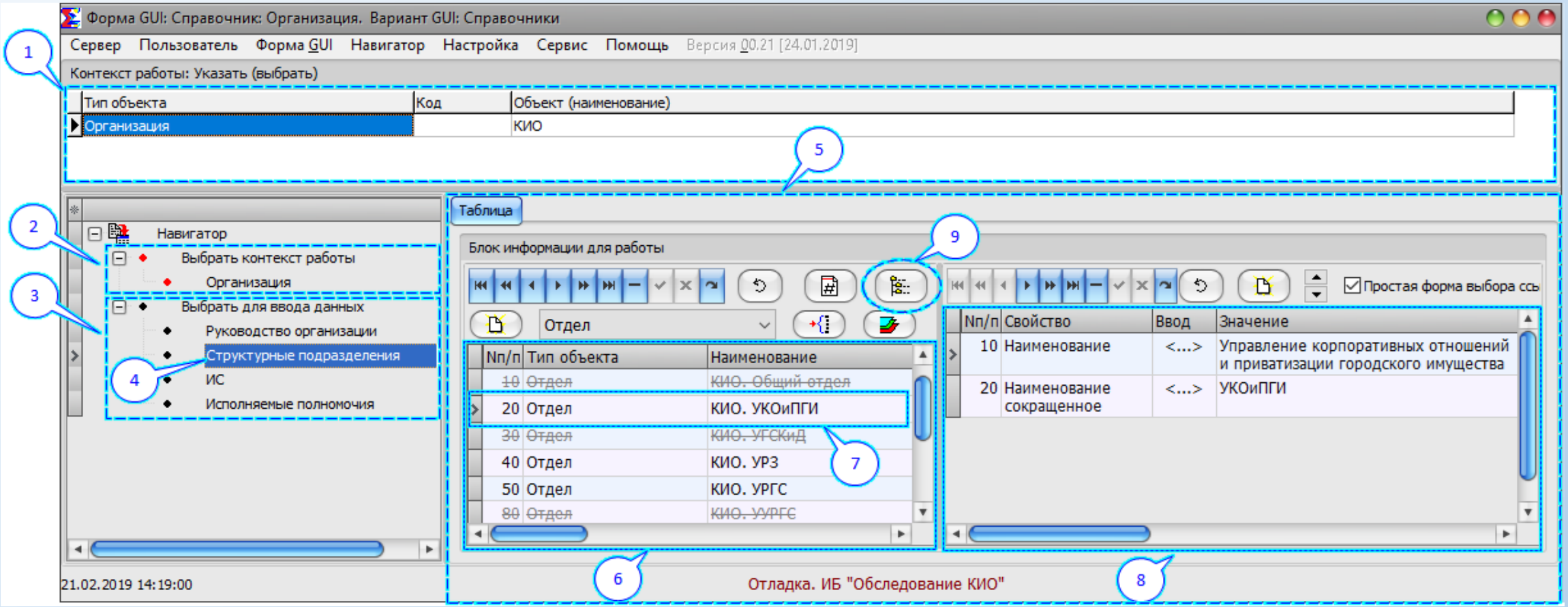
На рисунке ниже представлен пример, иллюстрирующий принципы интерактивной настройки GUI- объекта (типа «Меню»), на соответствующие объекты из обслуживаемой предметной области (так, как это реализовано в текущей версии).




К главному меню…

# GUI. ууФ. Практическая реализация. **Приложение ууФ**

На рисунке ниже представлен внешний вид ууФ при использовании варианта GUI «Справочник: Организация» (настройки см. на предыдущих слайдах).



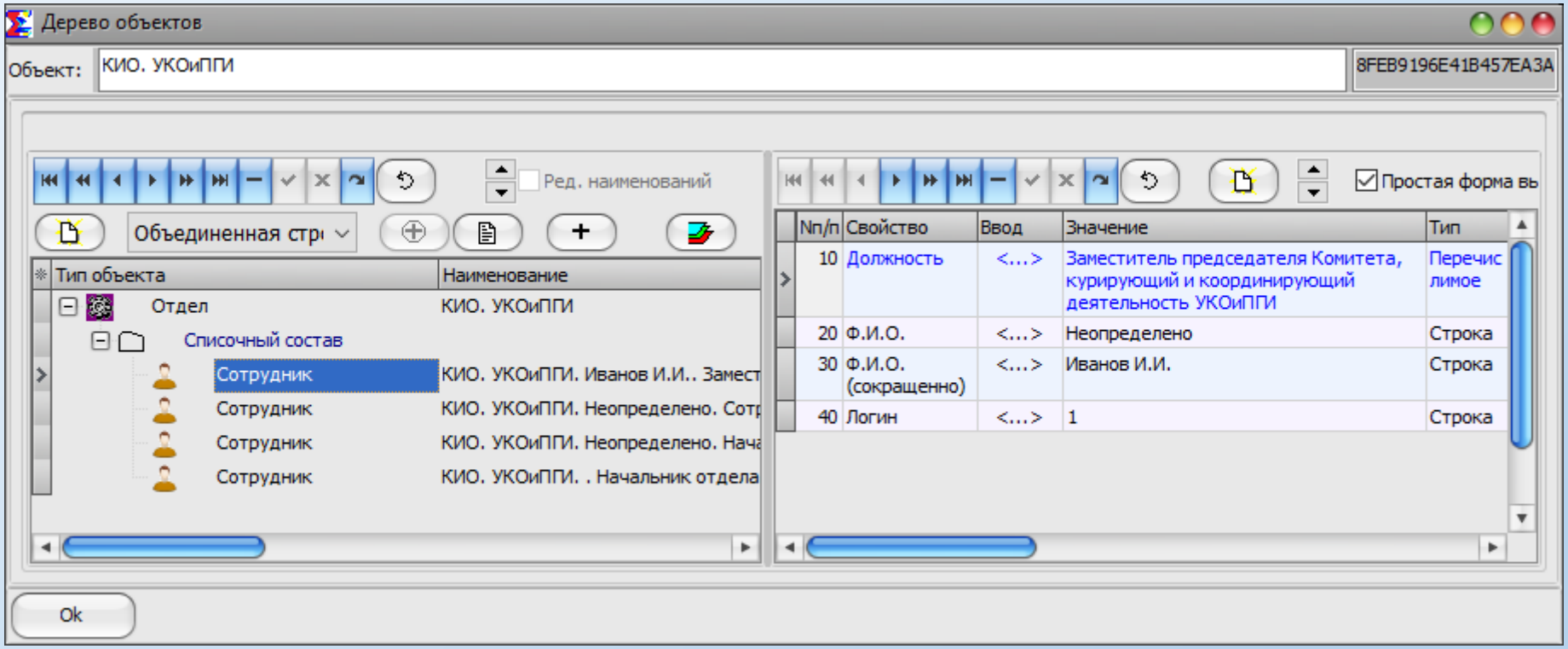

Выноски:

1 – контекст работы пользователя (область ЭУ, ориентированных на отображение и выбор контекста);
2 – контекст работы пользователя (область ЭУ, ориентированных на выбор контекста);
3 – область ЭУ, ориентированных на выбор соответствующего раздела информационной структуры объекта (из обслуживаемой предметной области);
4 – выбранный ЭУ (типа «Меню»);
5 – область ууФ, где размещены элементы управления, ориентированные на обеспечение процесса работы пользователя с данными из выбранного раздела;
6 – таблица объектов (из выбранного пользователем раздела);
7 – выбранный объект (в таблице объектов);
8 – таблица свойств выбранного объекта (в таблице объектов);
9 – ЭУ (кнопка), при нажатии на которую открывается полное дерево «потомков» выбранного объекта (в таблице объектов), см. пример далее.

К главному меню…

# GUI. ууФ. Практическая реализация. **Приложение ууФ**

Продолжение (см. предыдущий слайд).



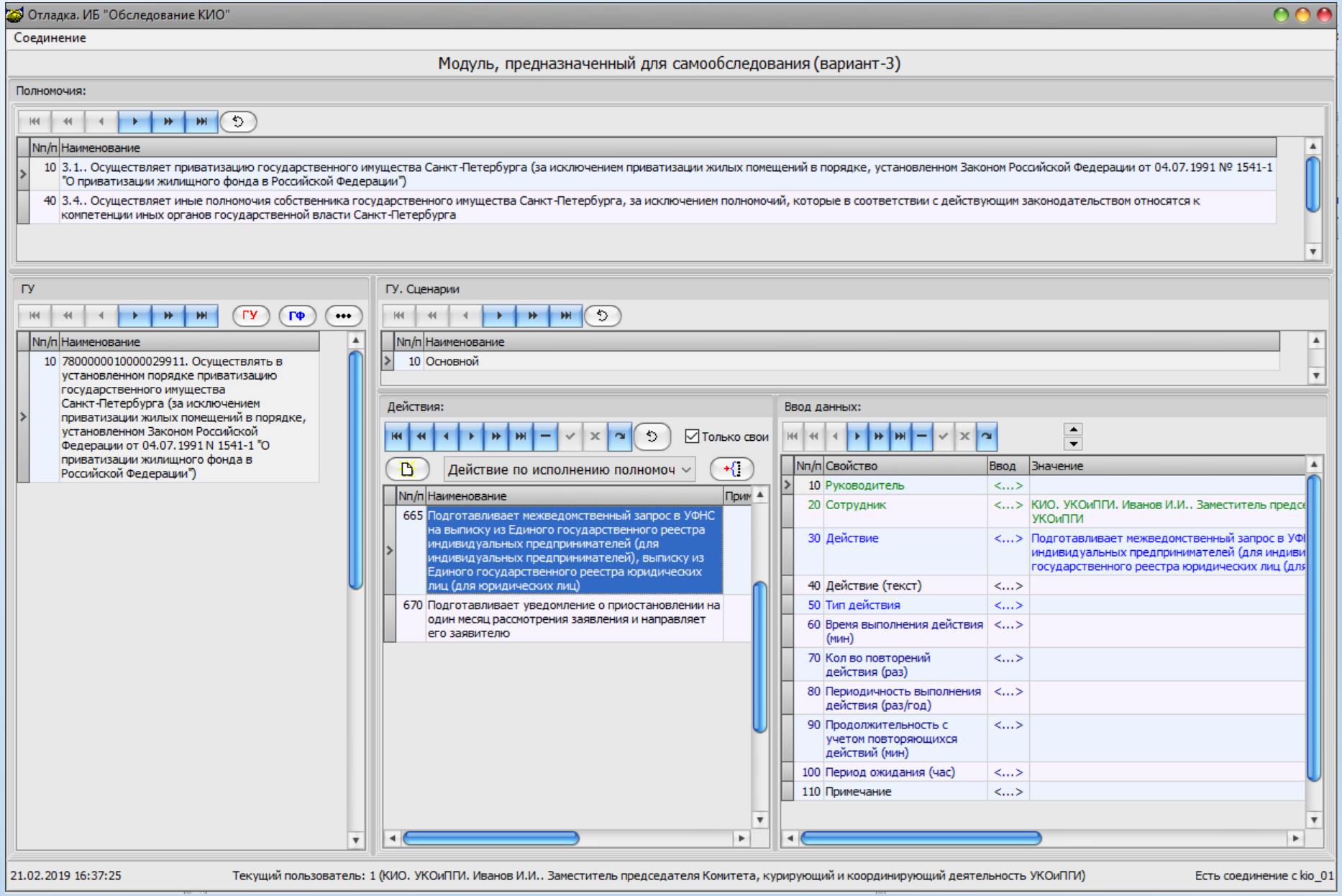

К главному меню…

# GUI. **Модуль самообследования**.

**Жесткий вариант реализации действующего макета приложения, предназначенного для проведения самообследования**

**Важно!**

Данный модуль разработан только для иллюстративных целей.
Его **<u>практическое применение не предусматривается</u>**.



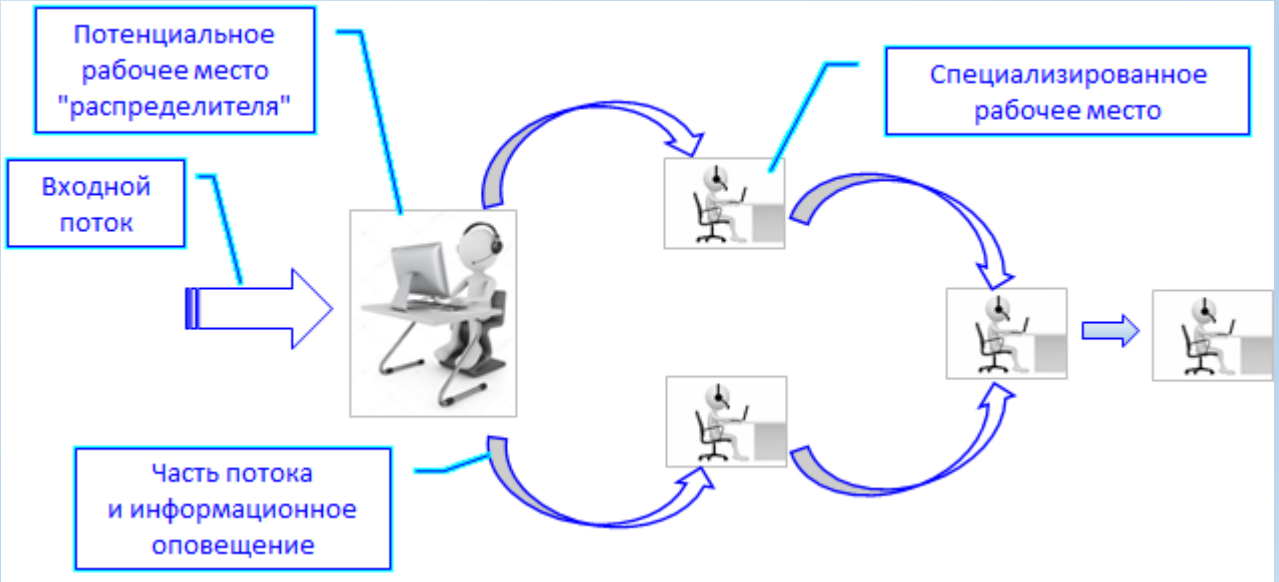

К главному меню…

# Технологическая цепочка распределенного ввода поступающей информации

**Пока что только на уровне декларации намерений:**

1. Конечному пользователю (уровень «Аналитик-администратор») должен быть предоставлен инструментарий, позволяющий оперативно формировать (в виде ориентированного графа) технологическую цепочку обработки входного информационного потока (с целью распределенного ввода данных в ИС) .
2. Информационный поток может быть разнородным (от одиночных и эпизодически поступающих партий документов, до непрерывно поступающих данных от иных ИС).
3. Каждый узел такого графа – это отдельное, специализированное рабочее место (включая и GUI, ест-нно).
4. Инструментарий должен предоставлять пользователю возможность оперативно и самостоятельно создавать (формировать) специализированные рабочие места.
5. Не должно быть никаких концептуальных ограничений на размер и сложность графа, а также, на количество и «свойства» специализированных рабочих мест.
6. Должна быть предусмотрена система информационного оповещения каждого специализированного рабочего места о готовности очередного информационного блока для ввода данных в ИС.
7. Не должно быть концептуальных ограничений на количество таких технологических цепочек.
8. Не должно быть никаких концептуальных ограничений для пользователя - оперативно производить модификацию любой технологической цепочки в зависимости от текущих условий эксплуатации.
9. Не должно быть никаких концептуальных ограничений на зависимость этого инструментария от любой предметной области.
10. Просматривается, также (но не факт), потенциальная необходимость разработки узко специализированного рабочего места для «распределителя работ» на «входе» поступающего информационного потока.

Иллюстрирующую схему см. на рисунке.




**Важно!**

Очевидно, что без **ууФ** (или иного, подобного инструментария) вообще нет смысла серьезно подходить к данному вопросу.

ууФ – унифицированная, управляемая GUI-форма.
Именно она (или нечто подобное) может быть основой возможности создания конечным пользователем «специализированных рабочих мест».
А без них сама цель – теряет знАчимость.

См., также, «Рабочий список пользователя»

К главному меню…

# Поиск информации

Назначение: поддержка механизма выявления взаимодействующих объектов при обработке текстов (документов) на естественном (русском) языке.

В процессе проработки вопроса были определены правила:

1. Тексты документов на естественном языке должны храниться в виде значений свойств (Текст, код=7) объектов в контексте объектной модели ИС. Т.е., предварительно тексты документов должны быть импортированы соответствующим образом.
2. Импортированные тексты документов должны быть проиндексированы соответствующим образом (формирование специализированных полнотекстовых индексов (FTI)).
3. Механизм поиска должен предоставлять конечному пользователю следующие возможности:
   3.1 Пошаговое, интерактивное формирование фильтров, включая следующие варианты:
      3.1.1 Сложные фильтры по значениям **не** текстовых свойств объектов.
      3.1.2 Сложные фильтры (с использованием FTI) по значениям текстовых свойств объектов.
   3.2 Совместное использование обоих вариантов.
   3.3 Возможность сохранения любого сформированного фильтра для дальнейшего использования.
   3.4 Возможность применения любого фильтра в любом режиме (пошаговый или непрерывный).
   3.5 Возможность применения определенного набора фильтров (из числа сформированных) в любом их сочетании с целью пакетной обработки «потока текстов».

К главному меню…

# Поиск информации

Определенные работы в данном направлении были проведены:

1. Сформированы (на уровне PostgreSQL-сервера) соответствующие таблицы в БД (схема qu), ориентированные на поддержку механизма пошагового поиска (фильтрации).
2. Сформированы (на уровне PostgreSQL-сервера) соответствующие таблицы в БД (схема si) и соответствующие ХФ, ориентированные на поддержку механизма индексации (FTI) объектов (в объектной модели), интерпретируемых, как документы.
   При этом (из-за сложности задачи), пришлось создать полностью собственный механизм построения FTI, включая и стеммер Портера на стороне клиентского ПО (версия алгоритма: для pascal).
3. Для тестирования поискового механизма был сформирован некий, существенно ограниченный по возможностям, прототип GUI (включая и конструктор поисковых запросов).
4. На зачаточном уровне механизм (поиск с использованием FTI) был протестирован.
5. По результатам можно судить, что его использование, как одного из инструментов для решения задачи, вполне возможно.

**Важно!**

1. Реализация стеммера Портера может быть зависима от языков программирования (со всеми вытекающими).
2. Полноценная проработка всего механизма поиска (а также его реализация на уровне GUI) **не** завершена.

К главному меню…

# Поиск информации. **Практическая реализация**

Ниже – скрины GUI, который использовался при тестировании.

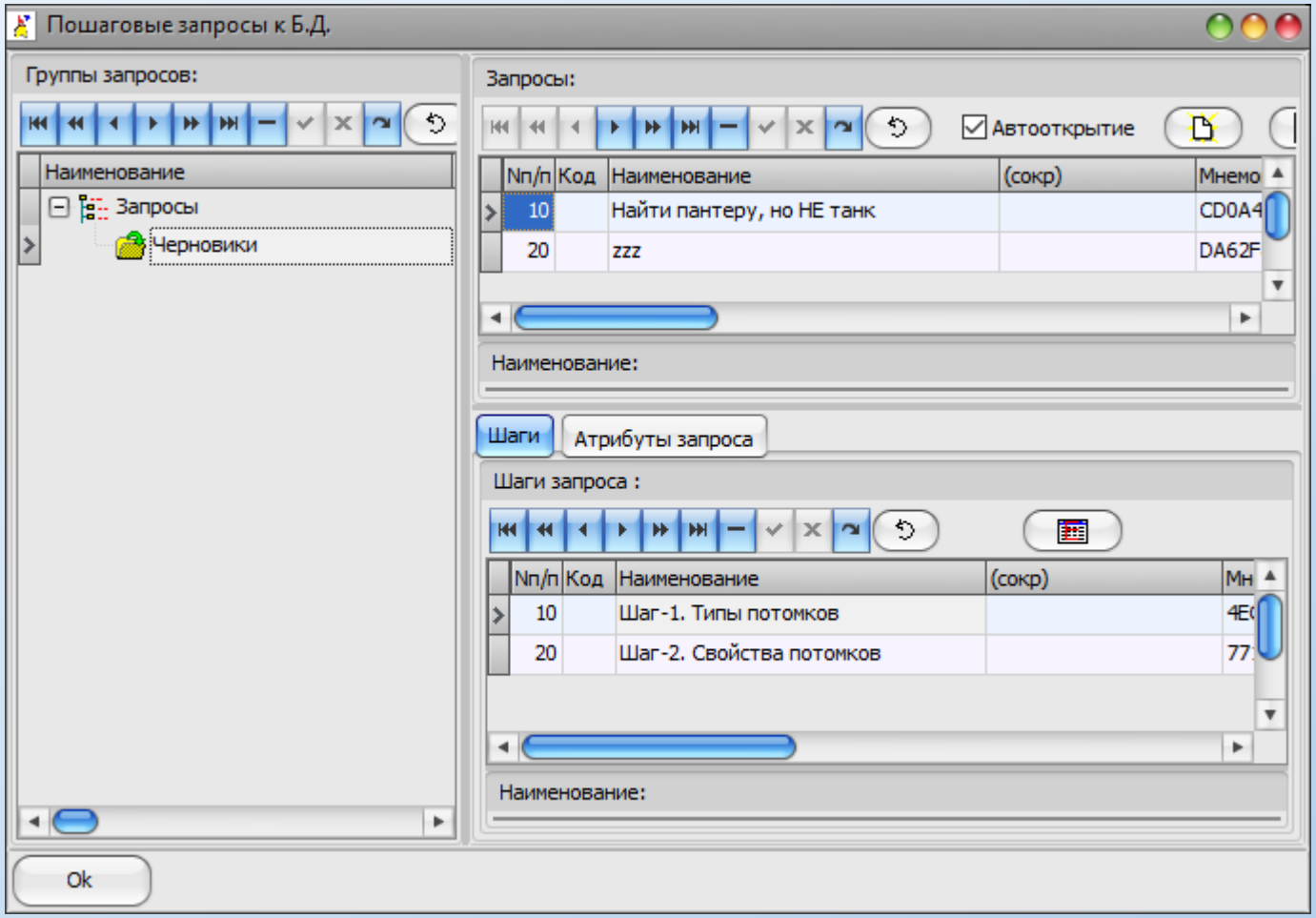



Рисунок 1 – Множество поисковых запросов

К главному меню…

# Поиск информации. Практическая реализация

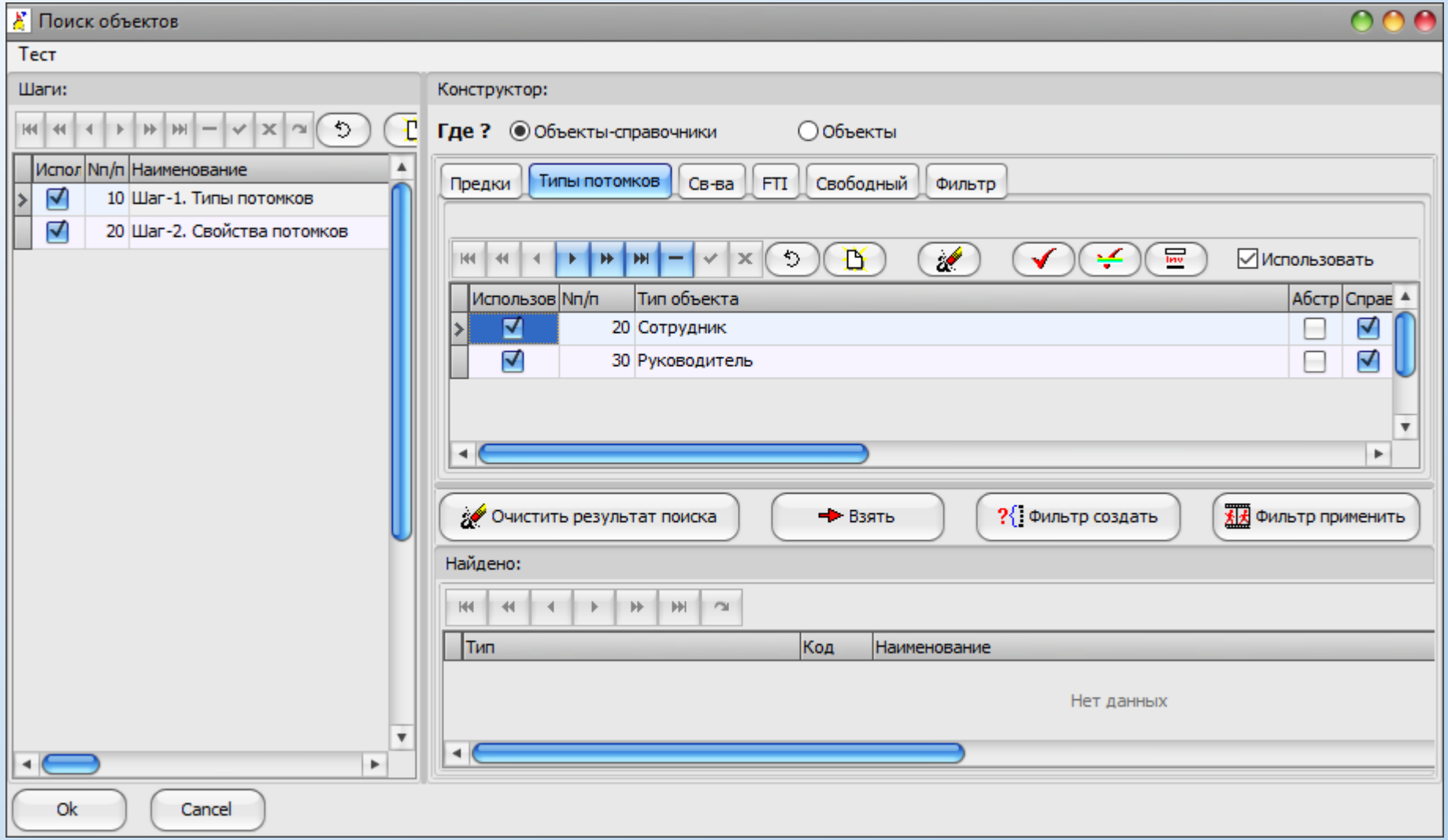



Рисунок 2 – Конструктор поисковых запросов

К главному меню…

# Обработка текстов на естественном языке

Задача - анализ текстов полномочий и возможно иных документов с целью выявления физических объектов, над которыми в рамках этих полномочий осуществляются какие-то действия (учет, мониторинг и др).

Конкретизация (в моей интерпретации):

1. Есть поток документов: тексты документов полномочий, гос.услуг и гос.функций (регламенты и др.); уставы предприятий; должностные инструкции; НПА; законы, приказы и распоряжения федеративного и регионального (СПб) уровней; письма; и т.д.
2. Необходимо провести анализ этих документов с целью извлечения информации:
   Объект(ы) <-> Взаимодействие <-> Субъект(ы);
   Объект(ы) <-> Взаимодействие <-> Объект(ы)
3. Тип взаимодействия должен быть детализирован.
4. Объекты(субъекты) должны быть классифицированы (по типам) или (что лучше) идентифицированы (на уровне объектной модели предметной области).
5. Процесс должен быть в максимальной степени автоматизирован.

Вообще-то, это уже уровень интеллектуальных ИС (если вообще не ИИ) со всеми вытекающими… ☺

Тем не менее, задача показалась интересной и определенные работы в этом направлении были проведены (см. поиск информации).

**Результат:**

1. Наличие механизма поиска (фильтрации) по текстам документов – **не** достаточен для полноценного решения задачи.
2. Далее (дополнительно), необходимо использовать более мощные методы анализа выявленной информации на уровне (как варианты):
   экспертные системы (в частности, предполагалось использовать Байесовскую ЭС);
   нейронные сети (в частности, предполагалось использовать нейронную сеть Хопфилда).
3. Вот до экспертных систем и нейронных сетей руки вообще не дошли (хотя, некоторый опыт их использования есть)…

И, кстати, наличие ЭС или нейронной сети – это тоже далеко не решение задачи (на практике).
Поскольку их еще тренировать надо! А в наших условиях это практически не выполнимо.

К главному меню…

# Разделение доступа пользователей к функционалу ИС

Механизм разработан на примитивном уровне:

1. Существуют следующие таблицы БД
   dir.roles_list - перечень всех доступных ролей пользователей (каждая роль имеет уникальный мнемокод, см. рисунок 1). Перечень ролей должен формироваться аналитиком интерактивно;
   wu.users_list – перечень пользователей ИС (с привязкой каждого пользователя к конкретной роли). Перечень пользователей формируется аналитиком интерактивно.
2. В объектную модель ИС добавлен раздел «Функционал ИС» (см. рисунок 2).
   Т.е., данный раздел (включая и его доступность для ролей пользователей) формируется интерактивно Аналитиком, по мере необходимости (в процессе наращивания возможностей ИС).
3. На уровне PostgreSQL-сервера создана ХФ, предоставляющая возможность (на уровне клиентского приложения) оценить доступность заданного функционала (по его коду) для заданной роли пользователей.
4. Алгоритм приведен на рисунке 3.
5. **Важно!** Механизм разработан, но **не** применен на уровне макетов наших программ.

Мнемокоды

| id_guid [PK] | mnemocode t_mnemo | code t_code | code_ob t_code | npp integer | name_max t_namemax |
|---|---|---|---|---|---|
| 793 F500 | SuperAdmin | '' | '' | 10 | СуперАдмин |
| ED7 F0BAC | Аналитик-Администратор | '' | '' | 20 | Аналитик-Администратор |
| 9EB4ED627C81409394C3F443C10998A9 | Пользователь | '' | '' | 30 | Пользователь |
| B525B5C0FE1442DBABD6E5472AD556A3 | Гость | '' | '' | 40 | Гость |

Рисунок 1 – Перечень ролей пользователей

К главному меню…

# Разделение доступа пользователей к функционалу ИС

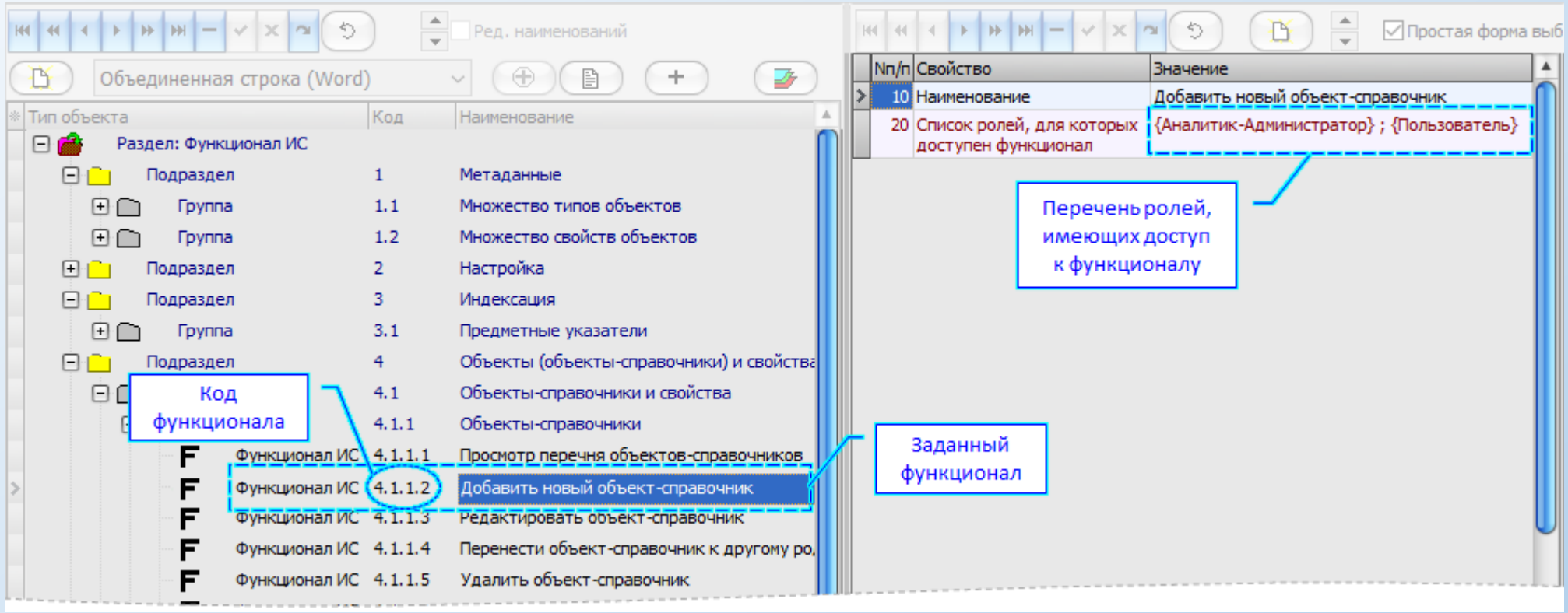



Рисунок 2 – «Раздел: Функционал ИС» в дереве объектов-справочников

Некоторый плюс данного подхода: кроме решения данной задачи (разделение доступа), параллельно формируется перечень всего функционала ИС (на уровне объектной модели ИС).

К главному меню…

# Разделение доступа пользователей к функционалу ИС

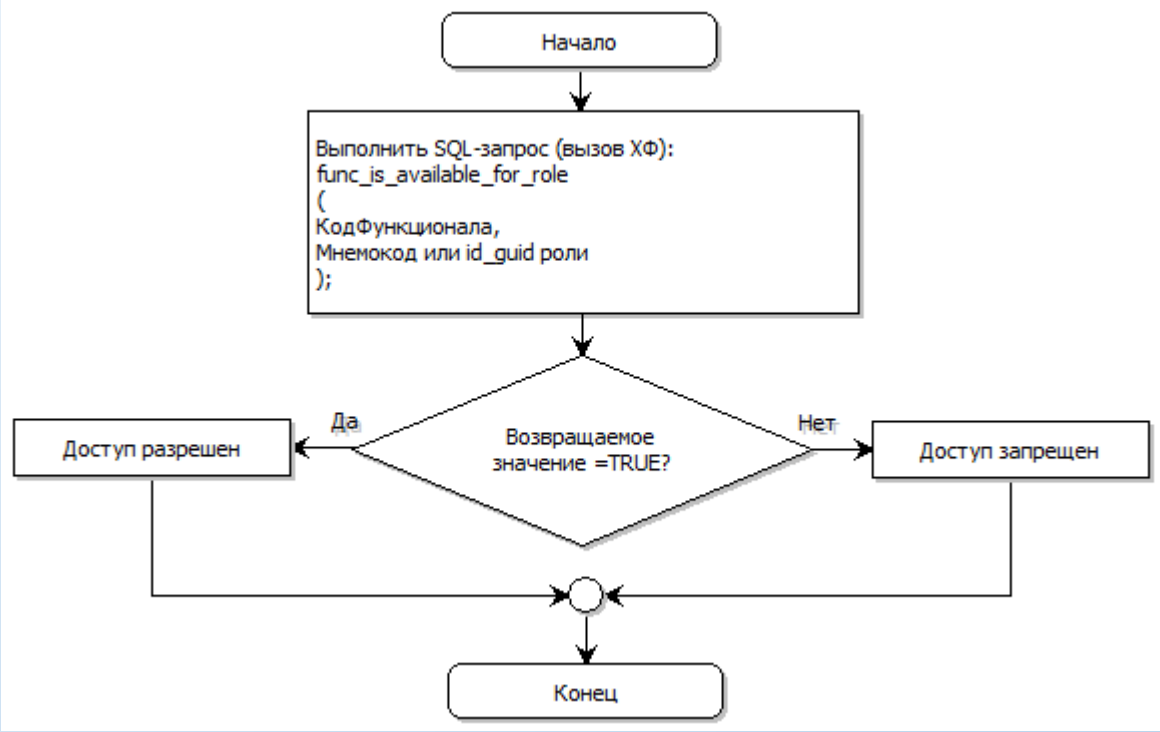



Рисунок 3 – Алгоритм оценки возможности доступа к функционалу

К главному меню…

# Миграция данных

**Миграция данных** - процесс управляемого перемещения данных между информационными базами (вне зависимости, в том числе, от принадлежности их к узлу) с целью синхронизации данных.

**Контекст миграции данных** - объект-справочник или объект, являющийся предком всех иных объектов, «участвующих» в конкретной миграции данных (может быть, в частности, и «корень дерева»).

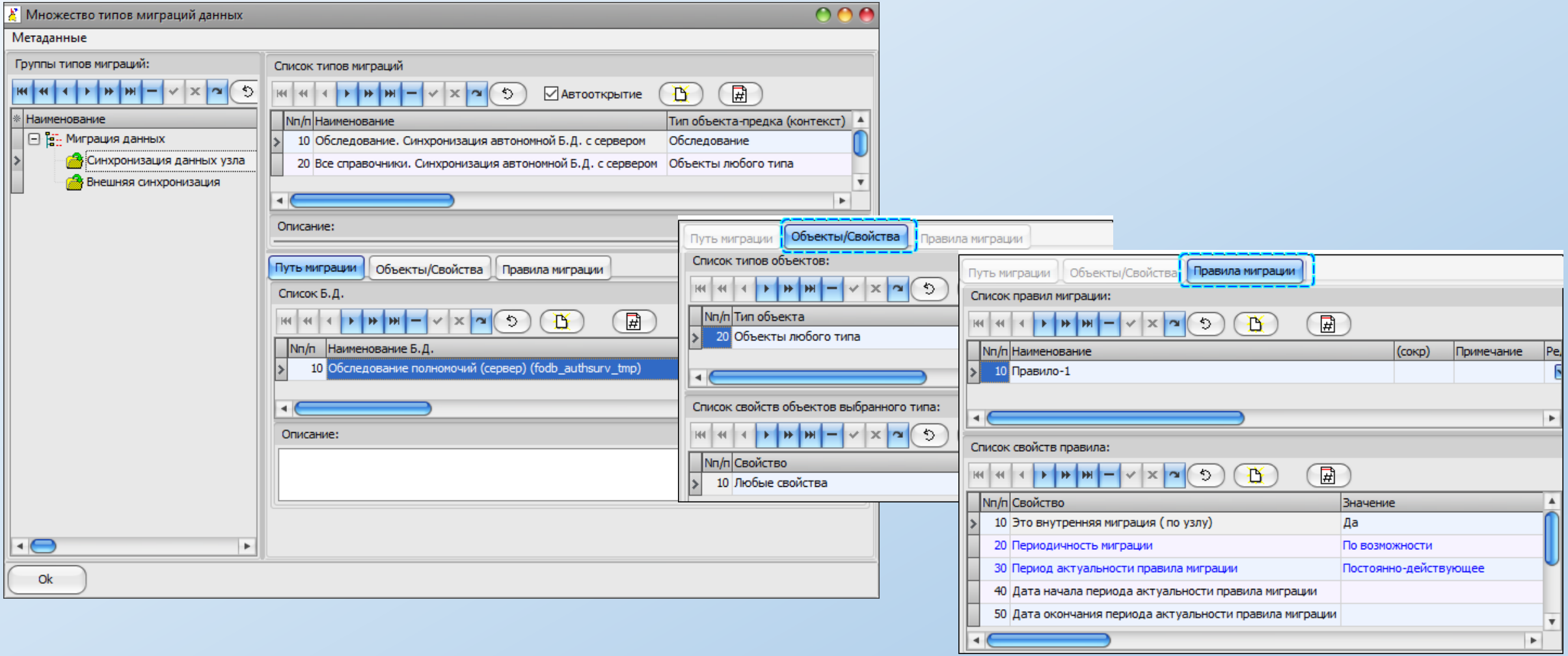
**Важно!**

1. Тема знАчимо сложная и требует детальной проработки.
2. В настоящий момент реализована только в части настроек (тестирование не проводилось).
3. Дальнейшая проработка вопроса осталась на уровне «мысленного эксперимента».
4. Причина банальна – физическое отсутствие времени.

К главному меню…

# Миграция данных. **Практическая реализация**

На рисунке ниже представлен внешний вид GUI, предназначенного для настройки миграции данных.




К главному меню…

# Иерархически распределенная ИС

**Назначение**:

1. Формирование информационных моделей (включая как информ. структуры, так и алгоритмы их обслуживания) по принципу "сверху-вниз". "Сверху-вниз" означает, что модель, сформированная на каком-либо узле "спускается" вниз (на след. уровень иерархии).
   При этом статус сформированных информационных моделей может быть следующим:
   - «глобальная модель» - означает, что свойства этой модели (информ. структуры и алгоритмы обслуживания) не могут быть изменены или отменены на уровне любых других узлов;
   - «наследуемая модель» - означает, что на любом следующем узле она может быть дополнена (но не отменена) в контексте как информ.структур, так и в контексте алгоритмов обслуживания;
   - «рекомендуемая модель» - означает, что на любом следующем узле она может быть заменена на другую или отменена вообще.
2. Формирование запросов на выборку и доставку информации по "иерархической лестнице" "снизу - в верх".

**Важно!** В настоящий момент все это пока что на уровне «декларации намерений».

К главному меню…

# Иерархически распределенная ИС

**База данных** – совокупность взаимосвязанных, хранящихся вместе данных при наличии такой минимальной избыточности, которая допускает их использование оптимальным образом для одного или нескольких приложений.

**Объект базы данных** – домены, таблицы, столбцы (поля) таблиц, индексы, триггеры, хранимые функции и т.д.

**Информационная база** – совокупность взаимосвязанных, хранящихся вместе данных об объектах (предметной области) и их свойствах, а также иных структурированных данных, которые используются в процессе обработки указанных выше данных.

**Семейство информационных баз** - множество информационных баз (вне зависимости от принадлежности их к узлу), имеющих одинаковое назначение (например: «обследование полномочий», «рецензирование документов», «рабочее место бухгалтера» и т.д.) и, соответственно, знАчимо совпадающие метаданные и информационные модели/структуры.

**Узел информационных баз** - одна из информационных баз, выполняющая роль узла для множества других информационных баз (см., также, таблицу dir.db_list).
Назначение:

- поддержка механизма миграции данных;
- поддержка общих справочников/классификаторов, используемых информационными базами узла;
- поддержка механизма интерактивного выбора пользователем соответствующей информационной базы узла для работы.

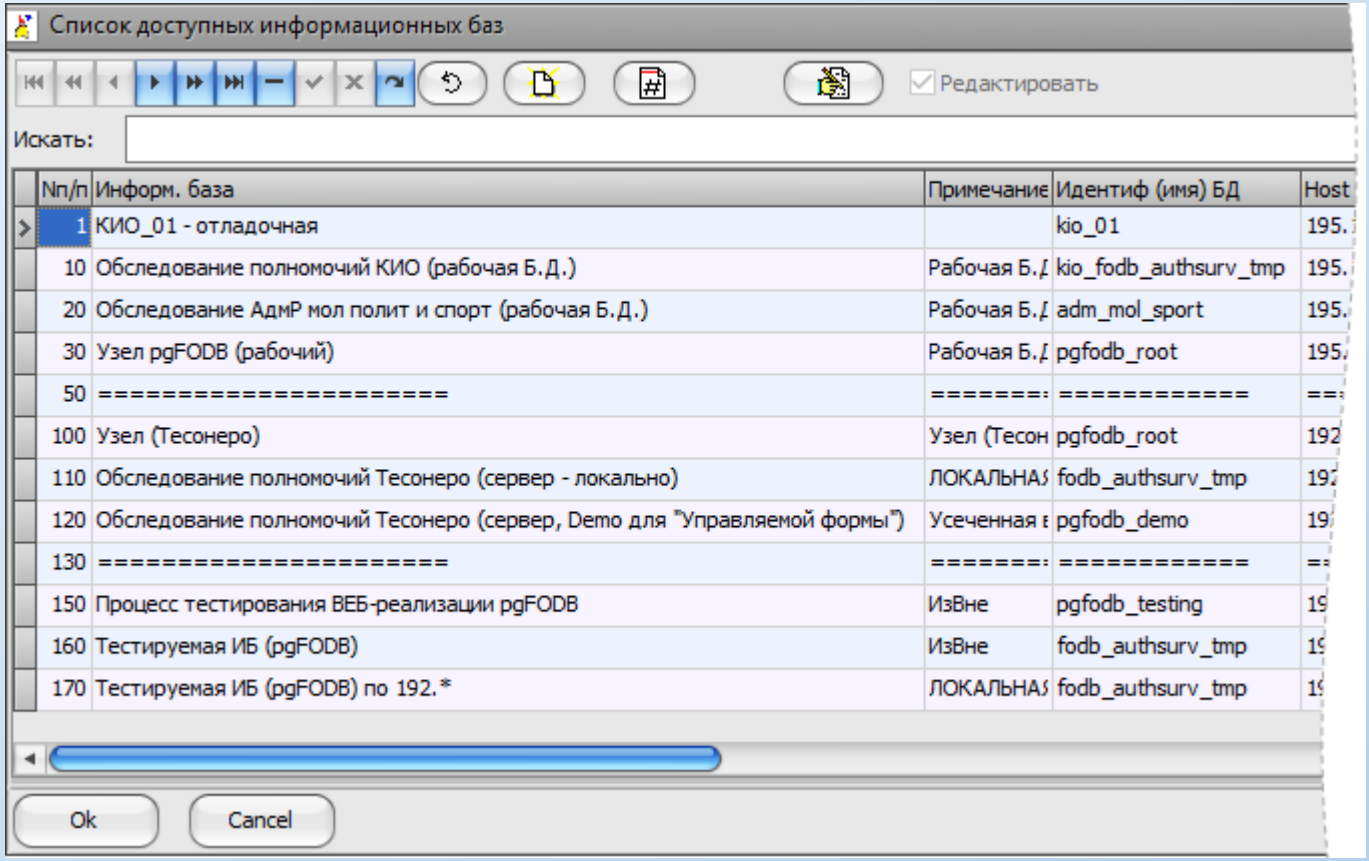
Примечание – любая ИБ может выполнять роль узла. Но лучше (для этих целей) использовать отдельную ИБ с именем (идентификатором) pgfodb_root.

**Важно!** В настоящий момент (в рамках узла) практически реализована только поддержка механизма интерактивного выбора пользователем соответствующей информационной базы узла для работы.

К главному меню…

# Иерархически распределенная ИС. Узел ИБ. **Практическая реализация**

На рисунке ниже представлен список ИБ узла.




| Nп/п | Информ. база | Примечание | Идентиф (имя) БД | Host |
|---|---|---|---|---|
| 1 | КИО_01 - отладочная | | kio_01 | 195. |
| 10 | Обследование полномочий КИО (рабочая Б.Д.) | Рабочая Б.Д | kio_fodb_authsurv_tmp | 195. |
| 20 | Обследование АдмР мол полит и спорт (рабочая Б.Д.) | Рабочая Б.Д | adm_mol_sport | 195. |
| 30 | Узел pgFODB (рабочий) | Рабочая Б.Д | pgfodb_root | 195. |
| 50 | ========== | ========= | ========== | == |
| 100 | Узел (Тесонеро) | Узел (Тесон | pgfodb_root | 192 |
| 110 | Обследование полномочий Тесонеро (сервер - локально) | ЛОКАЛЬНАЯ | fodb_authsurv_tmp | 192 |
| 120 | Обследование полномочий Тесонеро (сервер, Demo для "Управляемой формы") | Усеченная в | pgfodb_demo | 19 |
| 130 | ========== | ========= | ========== | == |
| 150 | Процесс тестирования ВЕБ-реализации pgFODB | ИзВне | pgfodb_testing | 19 |
| 160 | Тестируемая ИБ (pgFODB) | ИзВне | fodb_authsurv_tmp | 19 |
| 170 | Тестируемая ИБ (pgFODB) по 192.* | ЛОКАЛЬНАЯ | fodb_authsurv_tmp | 19 |

К главному меню…

# Межплатформенная реализация

Не будучи специалистом в «межплатформенности» - не берусь делать каких-либо знАчимых утверждений.

Тем не менее, есть некоторые вполне определенные моменты (применительно к Windows и к WEB):

1. Адаптивная ИС, при всей кажущейся «простоте и анархичности», является достаточно сложным, точным и сбалансированным решением. Это значит, что какие-либо не просчитанные «отклонения» от тех принципов, что в ней заложены, могут знАчимо дорого обойтись на каком-то этапе ее дальнейшего развития.
2. Разработка макетов приложений (с целью проработки и обкатки соответствующих методических и технических решений) производилась в среде Delphi 2009 (ОС Windows).
3. С целью унификации, однозначности исполнения, а также, упрощения реализации для разработчиков, существенная часть алгоритмов была перенесена на уровень PostgreSQL-сервер в виде ХФ (общее количество в пределах 600) и триггеров.
4. Как показал опыт (по реализации ИС в рамках WEB-версии) – почти все достаточно простые технические решения (легко и быстро выполненные в среде Delphi 2009) оказались знАчимо трудными при их реализации в WEB даже по готовым алгоритмам (прежде всего, по временнЫм затратам разработчиков).
5. Есть некоторые технические решения, которые очень маловероятно могут быть перенесены в WEB, но есть принципиальная возможность купирования этой проблемы (см. здесь).

**Важно!**

Как результат (см. выше):

1. В настоящий момент для ОС Windows у нас есть ряд готовых технических решений (на уровне **макетов** приложений), которые могут быть использованы для решения следующих практических задач:
   настройка ИС на предметную область (без ограничений);
   ввод и просмотр информации;
   обработка информации (хранящейся в ИС) с целью получения сложно-структурированных и сложно-форматированных отчетов (паспорт полномочия) при выводе их в MS WORD;
   формирование узла ИБ, где могут быть зарегистрированы N-е количество ИБ (без ограничений).
2. Вышеуказанные макеты приложений разработаны с учетом того, что работа конечного пользователя с ИС будет производиться в камеральных (а не в полевых) условиях и ориентированы на функционирование в рамках «Клиент-Серверной» архитектуры (т.е., при работе «из вне», при не очень хорошем канале связи, быстродействие может быть заметно снижено).
3. Часть технических решений (реализованных в рамках макетов приложений) предназначались для проработки и обкатки соответствующих методических решений и, соответственно, или не была завершена или не вышла на уровень полной готовности для практической работы.

К главному меню…

# Заключение. **Куда и как**

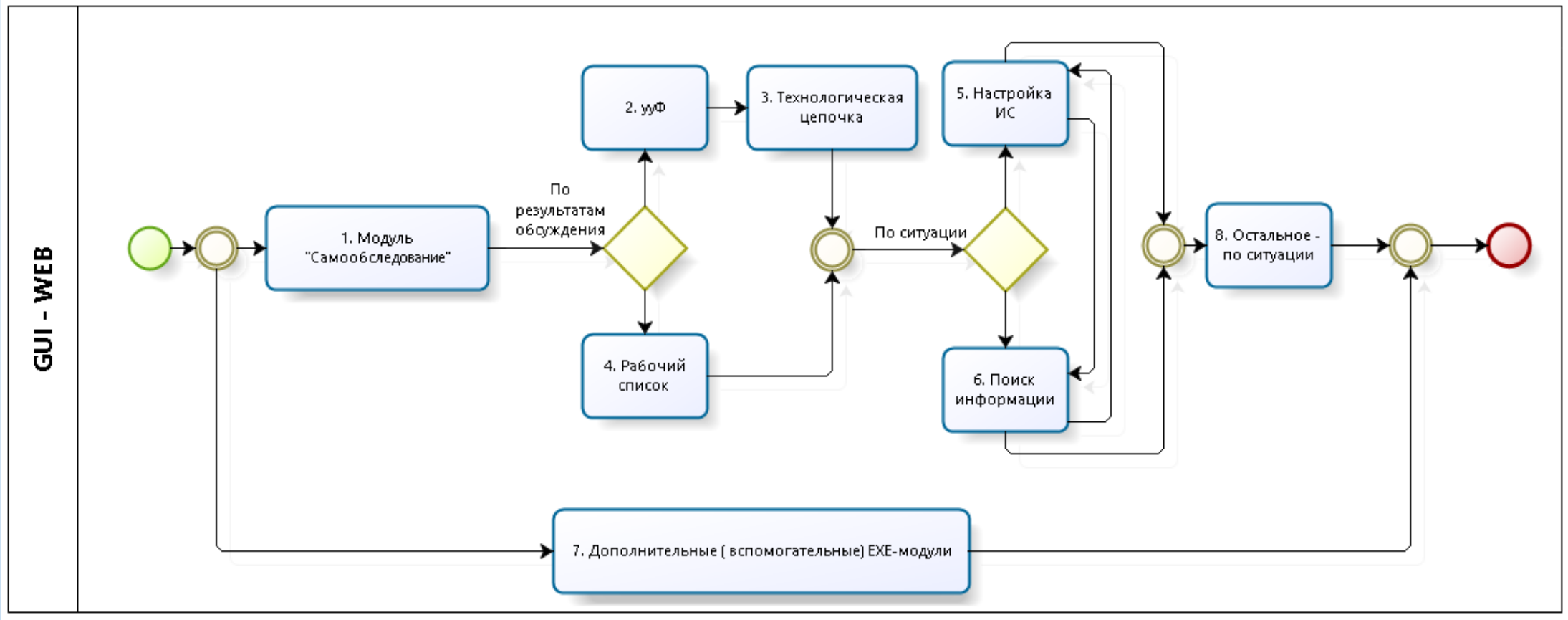
(в предположении, что конечная цель: реализация ИС в WEB-версии)

Последовательность разработки (как вариант):

1. Модуль самообследования (1);
2. Или ууФ (2) и, далее – технологические схемы (3);
3. Или «Рабочий список» (4);
4. Далее, по ситуации, или сначала «Настройка» (5), а потом «Поиск информации» (6) или наоборот;
5. Об «Остальном» (8) - пока рано загадывать.

Параллельно придется разрабатывать те EXE-модули (7), которые вряд ли удастся реализовать в WEB.

Под «Настройкой» понимается:
метаданные;
системные переменные;
предметные указатели;
прочее…




К главному меню…

# Конечная (в пределах разумного) цель

Некто или Нечто,
являющееся
носителем Интеллекта.
Озабочено своей ограниченностью
в плане "быстродействия"
и "объема памяти"

Желает получить "нечто",
что являлось бы неким
"расширителем" его
возможностей

К главному меню…

# Конечная (в пределах разумного) цель

Например, что-то вроде этого:


Выноски:

1 – некий модуль, являющийся интерфейсом между «условно мозгом» и «условно компом» (для нас он НеИнтересен).

2 – «Условно комп», содержащий этот самый «расширитель» возможностей и для нас это уже интересно. Поскольку это может быть та самая ИС, которая это все выполняет…

Закономерен вопрос: **Ну и какая она должна быть?**

Ответ очевиден:
**А такая же, как и собственная память (со всеми «прибамбасами») этого самого «Озабоченного».**
**НО! Очень быстрая, и без каких-либо ограничений на объем.**
**Т.е., это должно быть некое искусственное продолжение ЕГО памяти**

К главному меню…

# Конечная (в пределах разумного) цель

И все-таки. **Какая она должна быть?**
**Т.е., какая (и что это такое) память у Интеллекта (в исполнении Homo Sapiens)?**

Думаю, что до конца НИКТО (в настоящий момент) этого не знает…

Но можно предположить (прислушавшись к себе), что она (как минимум) должна быть:

- объектно-ориентированная (на предметную область);
- без концептуального ограничения на кол-во «обслуживаемых» предметных областей;
- адаптивная (ее можно оперативно «настроить» под любую предметную область);
- ассоциативная (это понятно);
- мульти-версионная и мульти-атрибутная (а поскольку мы, люди, сами версионны – никому из нас не дано знать исчерпывающе достоверно ВСЕ хотя бы о чем-то. Знают все и обо всем только Боги…);
- событийно-управляемая (это тоже понятно);
- имеет мощный поисковый механизм (не уступающий «оригиналу»);
- поддерживать миграцию данных (Input/Output) в рамках таких же «расширителей». И при чем, не только данных, но и данные о данных. Т.е., если Некто-1 знает о том, что Некто-2 владеет информацией о «чем-то», но это «что-то» для Некто-1 сейчас не нужно, то Некто-1 должен сформировать «нечто» в своей памяти об этом знании и автоматически ассоциировать «это» с «чем-то», уже имеющемся у него в памяти.
- и т.д.

К главному меню…

# Конечная (в пределах разумного) цель

Т.е., если **«нацелиться» на такой «расширитель»** и упорно идти в этом направлении по шагам (даже и не совсем по прямой), то мы и получим вот такую ИС:

FODB (далее – Система) разрабатывается, как объектно-ориентированная (на соответствующую предметную область), адаптивная, событийно-управляемая и иерархически распределенная ИС, предназначенная для автоматизации решения качественных задач без концептуальных ограничений на вид (тип) охватываемой предметной области.

Что мы, собственно, уже сейчас частично и имеем…

Главное – не терять конечную цель и не сваливаться в конъюнктурность.
Иначе это очень дорого обойдется…

Но вот этого мы **не** получим в данном варианте реализации:

- мульти-версионная и мульти-атрибутная;
- мультиязычная.

Поскольку я изначально был вынужден ориентироваться только на качественные задачи.
И теперь «дотягивание» этого варианта до более полноценного уже настолько дорого, что дешевле начать новую с нуля.

К главному меню…