СИСТЕМА БЫСТРОГО ПРОГРАММИРОВАНИЯ APAWE

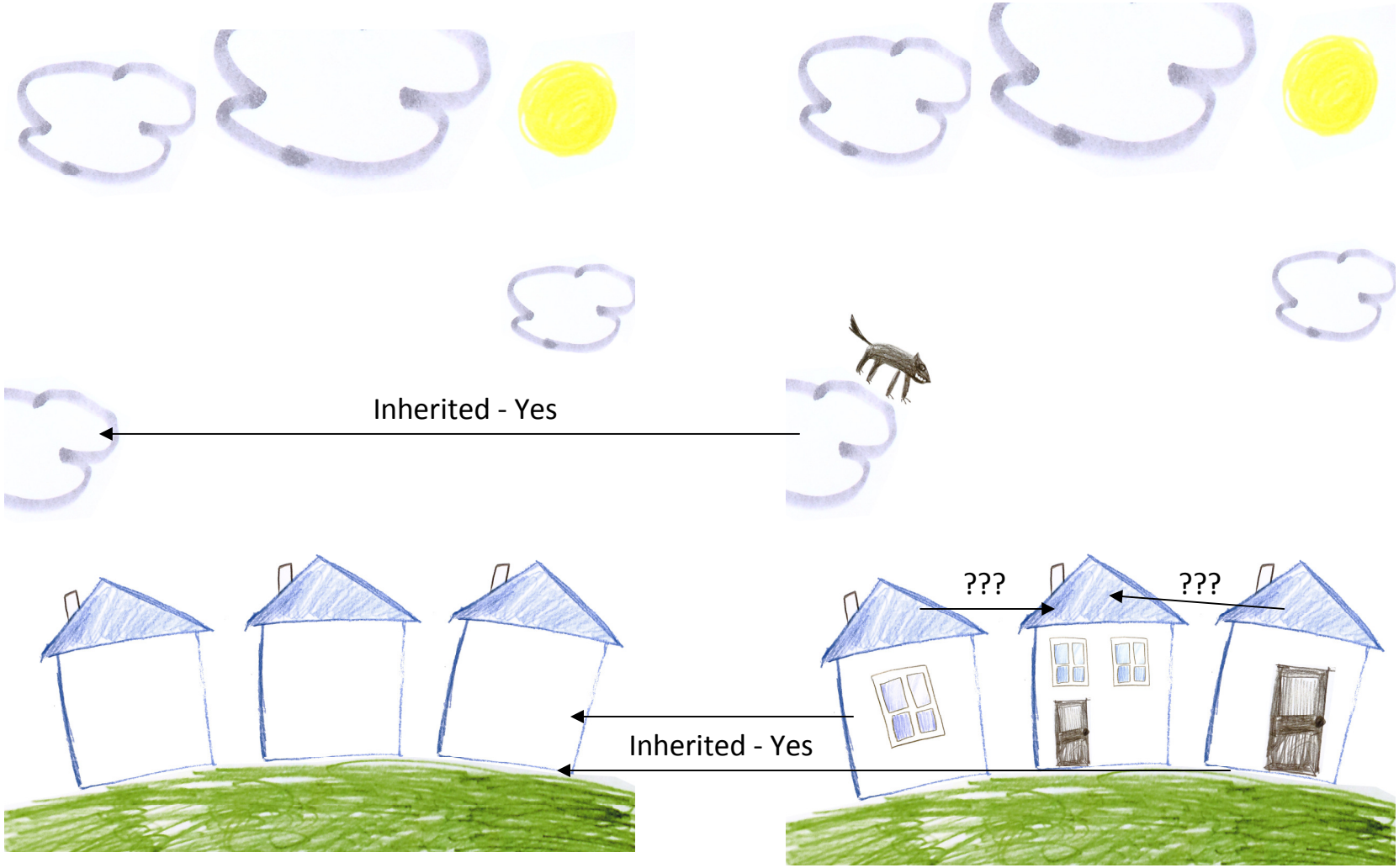
SoftLand Systems

Разработчик: Дмитрий Кожевин

Докладчик: Андрей Апаринов

Октябрь, 2011

НЕДОСТАТКИ ТРАДИЦИОННОГО ПОДХОДА



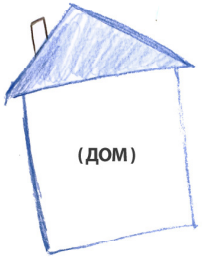
НЕ НУЖНО ПИСАТЬ ПРОГРАММУ, ДОСТАТОЧНО ЕЕ СОБРАТЬ

Элементы для сборки

(ОБЛАКО)

(СОЛНЦЕ)

(ДВЕРЬ)



(ДОМ)

(ОКНО)

(БУДКА)

(ПЁС)

(КОТ)

Вариант сборки 1

Вариант сборки 2

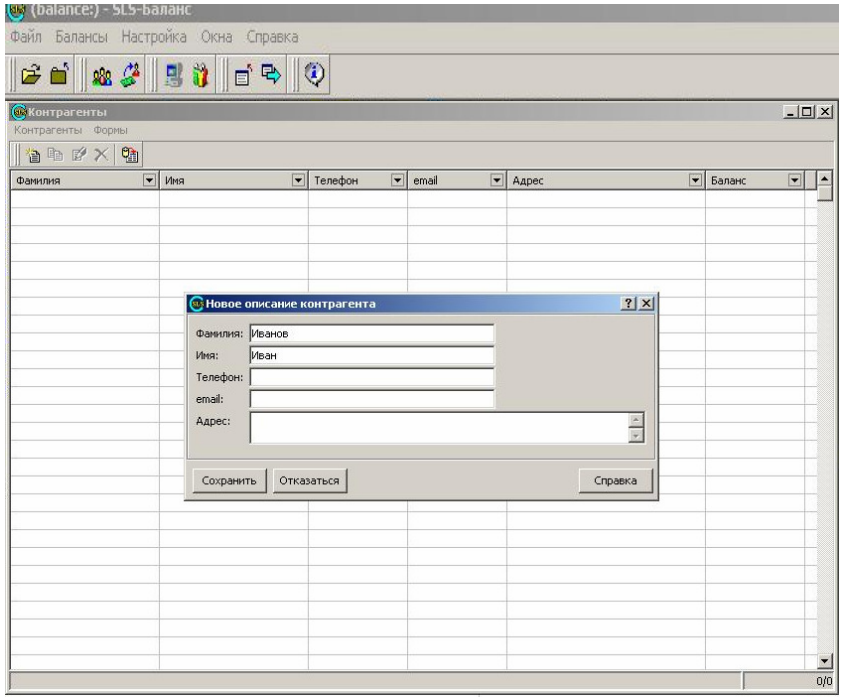
ХОТИМ РАЗРАБОТАТЬ ПРОГРАММУ

Что хотим получить

Электронную БД для записи долгов, которая позволяет:

1. Ввести и сохранить Фамилию, Имя, mail, адрес и телефон контрагента;
2. Отмечать платежи выданные и полученные;
3. Считать баланс на дату.

Примерный вид результата



КАК ВЫГЛЯДИТ КОД В ТЕХНОЛОГИИ APAWE

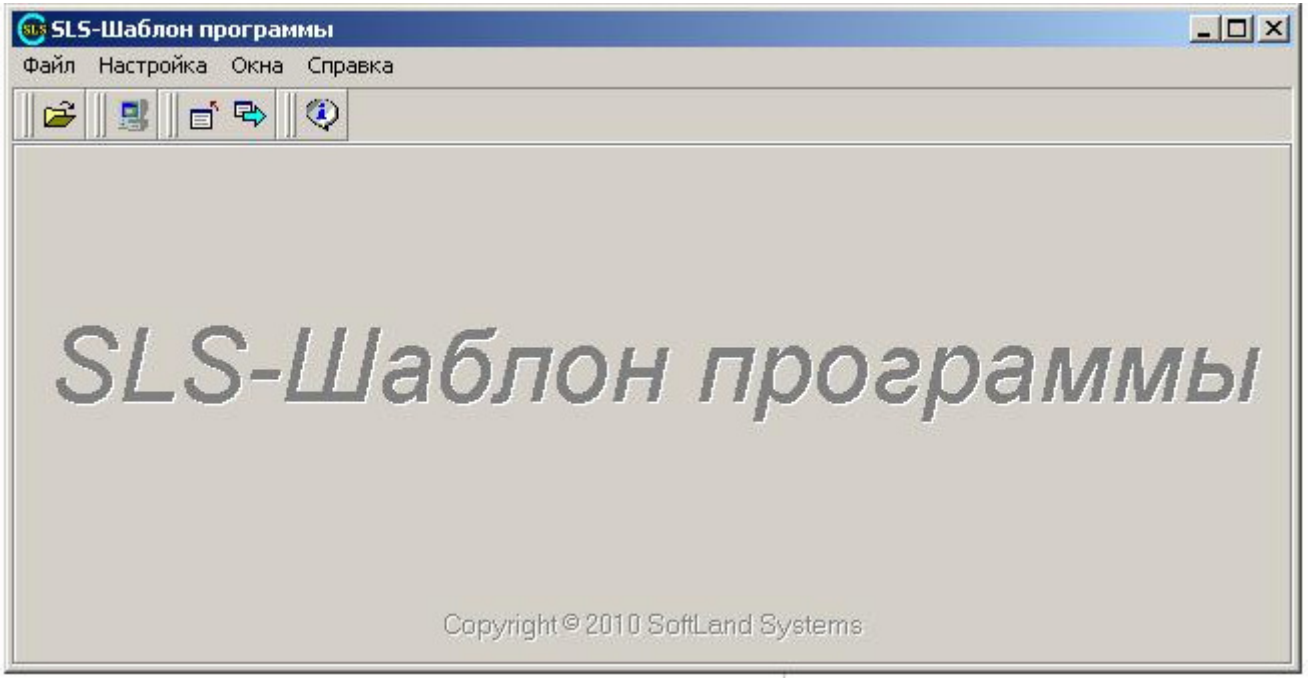
```
unit uProgramParts;

interface

uses
  uProgramPart, uDescription;

const
  SCaption = 'SLS-Баланс';
  SAppTitle = 'Система управления долгами';
  SVersion = 'Версия 0.0 для MS SQL Server';
  STitle = 'SLS-Баланс';
  ExeFile: string = 'balance.exe';

  SBalanceMenu = 'Балансы';
  SContractors = 'Контрагенты';
  SPayments = 'Платежи';

// Главное меню
  PPMain: TProgramPartDef = (Caption: ''; Image: -1; Name: 'Main');
    PPBalanceMenu: TProgramPartDef = (Auto: True; Caption: SBalanceMenu; Image: -1; Name: 'BalanceMenu');
// Контрагенты
      PPContractors: TProgramPartDef = (Auto: True; AutoEdit: True; Caption: SContractors; DataName: 'Contractor'; FF: True; ForChoose: True; Hei
// Платежи
      PPPayments: TProgramPartDef = (Auto: True; AutoEdit: True; Caption: SPayments; DataName: 'Payment'; Height: 600; Image: 218; Name: 'Payment

procedure SetProgramLinks;

implementation

procedure SetProgramLinks;
begin
  RegisterProgramLinks([
    PL(PPMain, PPBalanceMenu, [ppaShow]),
      PL(PPBalanceMenu, PPContractors, [ppaShowAsTable]),
      PL(PPBalanceMenu, PPPayments, [ppaShowAsTable])
  ]);
end;

end.
```

ПОЧТИ ВСЕ

ОБЩИЙ ПРОЦЕСС НАПИСАНИЯ ТАКОЙ ПРОГРАММЫ

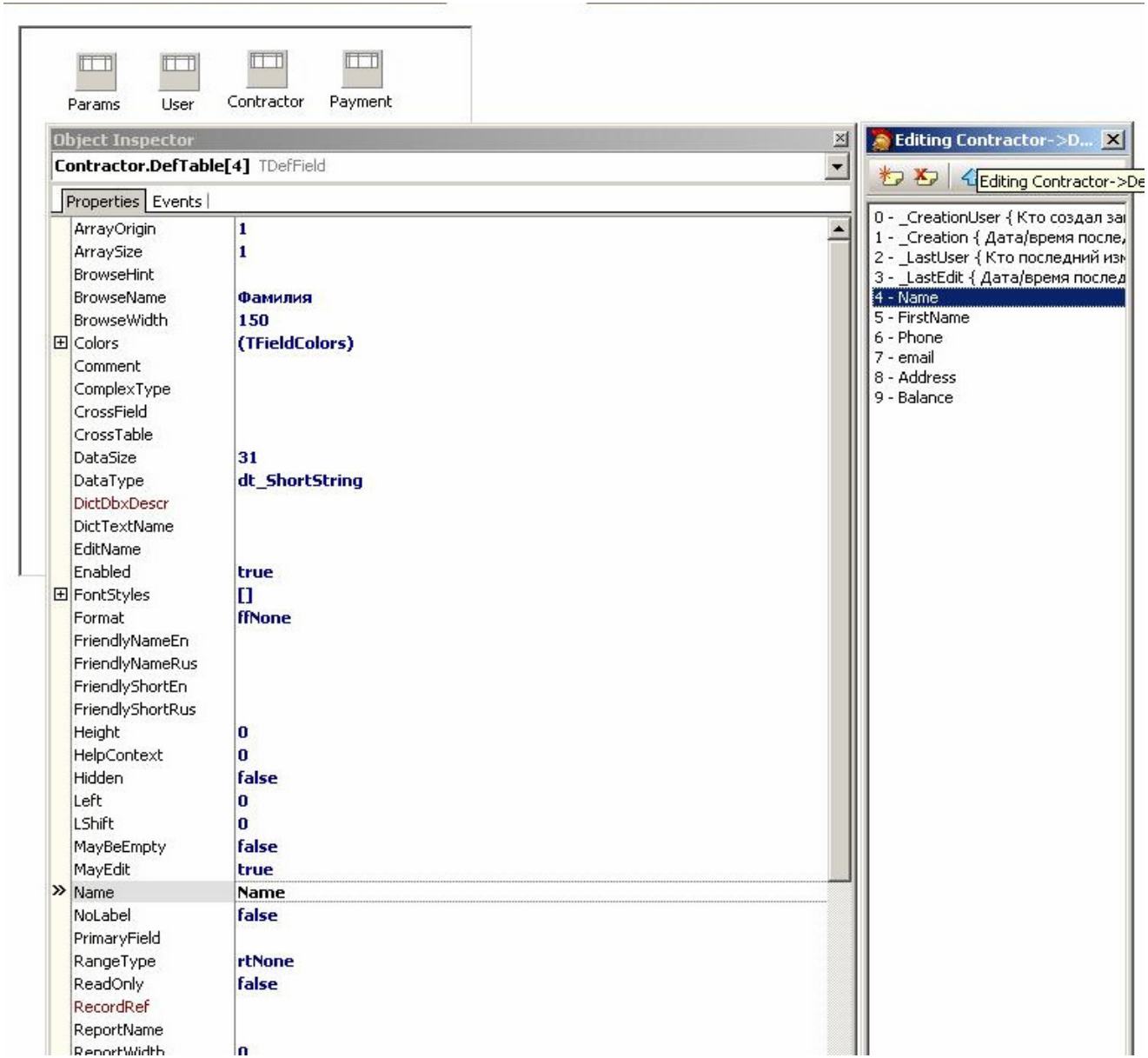
1. Взять шаблон «пустой программы»
2. Описать БД программы
3. Выбрать из каких элементов состоит программа
4. Установить взаимосвязи элементов
5. Запрограммировать нестандартную функциональность

ИСПОЛЬЗОВАНИЕ ШАБЛОНА ПУСТОЙ ПРОГРАММЫ



Шаблон подходит для любой программы. Дальше программируем отличия от шаблона.

ОПИСАНИЕ БД ПРОГРАММЫ



ОПИСАНИЕ ЭЛЕМЕНТОВ ПРОГРАММЫ

```
// Контрагенты
      PPContractors: TProgramPartDef = (Auto: True; AutoEdit: True; Caption: SContractors; DataName: 'Contractor'; FF: True; ForChoose: True; Hei
// Платежи
      PPPayments: TProgramPartDef = (Auto: True; AutoEdit: True; Caption: SPayments; DataName: 'Payment'; Height: 600; Image: 218; Name: 'Payment
```

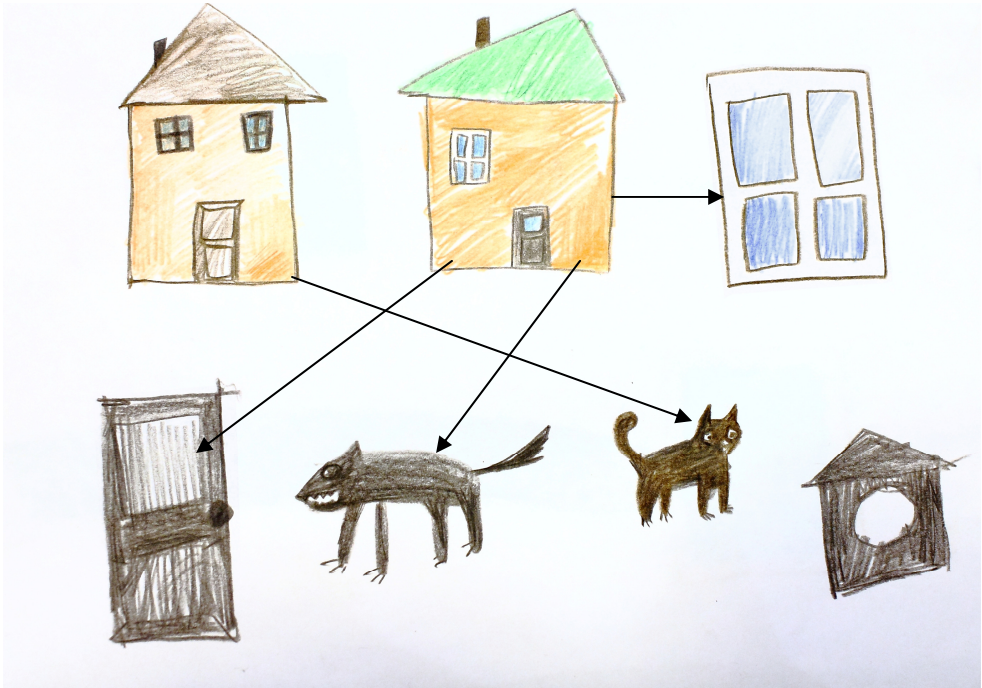
Программное определение **TProgramPartDef** позволяет определять элементы интерфейса программы путем простого перечисления их свойств.

*Программное определение (**ProgramDef**)* является основной информационной единицей технологии APAWE. Его назначение аналогично роли ДНК в живых организмах. Варьируя количество хромосом и их «параметры», можно получить почти что угодно.

ОПИСАНИЕ ВЗАИМОСВЯЗЕЙ ЭЛЕМЕНТОВ ПРОГРАММЫ

```
procedure SetProgramLinks;
begin
  RegisterProgramLinks([
    PL(PPMain, PPBalanceMenu, [ppaShow]),
      PL(PPBalanceMenu, PPContractors, [ppaShowAsTable]),
      PL(PPBalanceMenu, PPPayments, [ppaShowAsTable])
  ]);
end;
```

Чтобы программа начала работать, достаточно установить взаимосвязи созданных элементов интерфейса.



ПРОГРАММИРОВАНИЕ ОТЛИЧИЙ - 1

```
type
  TEditContractor = class(TDbxDialogForm)
  private
  protected
  public
    class function CheckDelCond(Sender: TObject; iAdr: longint; bSay: boolean): boolean; override;
    class function DelDependedObjects(Sender: TObject; iAdr: longint; bBefore: boolean; CRec: pointer): boolean; override;
  end;

implementation

uses
  uLogic;

class function TEditContractor.CheckDelCond(Sender: TObject; iAdr: Integer;
  bSay: boolean): boolean;
var
  Balance: extended;
begin
  Result := abs(GetAsFloat('contractor', 'balance', iAdr)) < MinReal;
  if not Result and bSay then
    ErrorMess('Нельзя удалить контрагента с ненулевым балансом.');
end;

class function TEditContractor.DelDependedObjects(Sender: TObject; iAdr: Integer;
  bBefore: boolean; CRec: pointer): boolean;
begin
  Result := True;
  if not bBefore then
    exit;
  DelOnName('payment', 'contractor', iAdr);
end;

initialization

  RegisterClass(TEditContractor);

end.
```

ПРОГРАММИРОВАНИЕ ОТЛИЧИЙ - 2

```
uses
  uLogic;

class function TEditPayment.DelDependedObjects(Sender: TObject; iAdr: Integer;
  bBefore: boolean; CRec: pointer): boolean;
begin
  Result := True;
  if bBefore then
    exit;
  with TPayment(CRec^) do
    ChangeExtendedField('contractor', 'balance', Contractor, Summa, 0);
end;

function TEditPayment.DoAdd: longint;
begin
  Result := inherited DoAdd;
  if Result > 0 then
  with TPayment(DbxData.PCRec^) do
    ChangeExtendedField('contractor', 'balance', Contractor, 0, Summa);
end;

function TEditPayment.DoEdit: longint;
var
  CPaymentOld: TPayment;
begin
  ReadOnAdr(Description.Payment, FAdr, CPaymentOld);
  Result := inherited DoEdit;
  if Result > 0 then
  begin
    with CPaymentOld do
      ChangeExtendedField('contractor', 'balance', Contractor, Summa, 0);
    with TPayment(DbxData.PCRec^) do
      ChangeExtendedField('contractor', 'balance', Contractor, 0, Summa);
  end;
end;

class procedure TEditPayment.DoTotalRefresh(Sender: TObject; AGroup: boolean);
begin
  if Sender is TProgramBasis then
    with Sender as TProgramBasis, Description do
      TotalRefresh(False, [Payment, Contractor]);
end;
```

Программируем только то,

что характерно

только для этого

раздела программы.

ЧТО ВХОДИТ В ТЕХНОЛОГИЮ APAWE

*Программное определение (**ProgramDef**)* является основной информационной единицей технологии APAWE. Его назначение аналогично роли ДНК в живых организмах. Варьируя количество хромосом и их «параметры», можно получить почти что угодно.

Термин «часть программы» (***ProgramPart***) используется для описания таких частей приложения, как пункты главного меню главной формы, а также для всех других частей, имеющих собственную форму.

Термин «узел программы» (***ProgramNode***) используется для описания таких компонентов форм, как закладка, узел дерева и т.п. ***ProgramNode*** не имеет собственной формы и предназначен для описания компонентов ***ProgramPart***.

Каждый элемент ***ProgramBasis*** может быть связан с таблицей данных.

Библиотека ***VisualLibrary*** представляет собой набор компонент для реализации интерфейсов, и аналогична стандартной библиотеке Delphi, с тем отличием, что для каждого компонента добавлены методы, позволяющие «автоматически» помещать его на создаваемые формы.

Процедура ***PL*** определяет взаимосвязи между элементами программы.

ПРЕИМУЩЕСТВА ТЕХНОЛОГИИ

1. Очень высокая скорость создания исходного кода для абсолютно новых приложений;
2. Простота отладки создаваемых приложений;
3. Простота программирования.

Простота тестирования определяется почти полной независимостью ветвей программы друг от друга.

Простота программирования достигается идеей технологии – нужно описать только то, что хочется получить в результате, при этом, практически не заботясь об алгоритме, который реализует конечную программу.

Использование технологии APAWE избавляет от массы рутинной работы и существенно упрощает и убыстряет разработку программы.

РЕАЛИЗОВАННЫЕ ПРОЕКТЫ

Система управления бизнес-процессами компании.

Срок разработки - 4 месяца.

Система автоматизации деятельности ветеринарной клиники.

Срок разработки - 7 месяцев.

О СИСТЕМЕ УПРАВЛЕНИЯ БИЗНЕС-ПРОЦЕССАМИ

Система позволяет:

- настроить логику бизнес-процессов;
- указать должности исполнителей;
- определить набор услуг, которые компания оказывает клиенту (если необходимо);
- задать регламентное время исполнения для каждого задания;
- задать связанные документы и привязать шаблоны, на основании которых документы создаются;
- автоматически формировать задания индивидуально для каждого сотрудника;
- контролировать состояния исполнения по любой задаче;
- отслеживать по каждому сотруднику исполненные, просроченные и задания в исполнении;
- контролировать расчеты с клиентами;
- разграничивать права доступа к информации.

ОПИСАНИЕ ЛОГИКИ БИЗНЕС-ПРОЦЕССОВ

Логика бизнес-процессов компании задается путем описания шаблонов.

Для задания шаблона бизнес-процесса используются понятия

- Шаг;
- Задание.

Предполагается, что шаблон бизнес-процесса определяется шагами, между которыми устанавливаются логические связи и должностями сотрудников, которые должны исполнять шаги бизнес-процесса. При этом каждый шаг может состоять из нескольких заданий, которые должны быть выполнены для завершения шага.

Каждый шаблон бизнес-процесса может иметь состояния:

- Проект – в процессе разработки и не может быть использован для отражения реальных бизнес-процессов;
- Готов – рабочий шаблон бизнес-процесса;
- Архив – отработавший шаблон бизнес-процесса, который более не используется компанией.
- Система обладает функцией проверки правильности задания шаблона бизнес-процесса и встроенным графическим редактором диаграмм бизнес-процессов.

ГРАФИЧЕСКОЕ ПРОЕКТИРОАНИЕ БИЗНЕС-ПРОЦЕССОВ

Система обладает функцией проверки правильности задания шаблона бизнес-процесса и встроенным графическим визуализатором диаграмм бизнес-процессов .

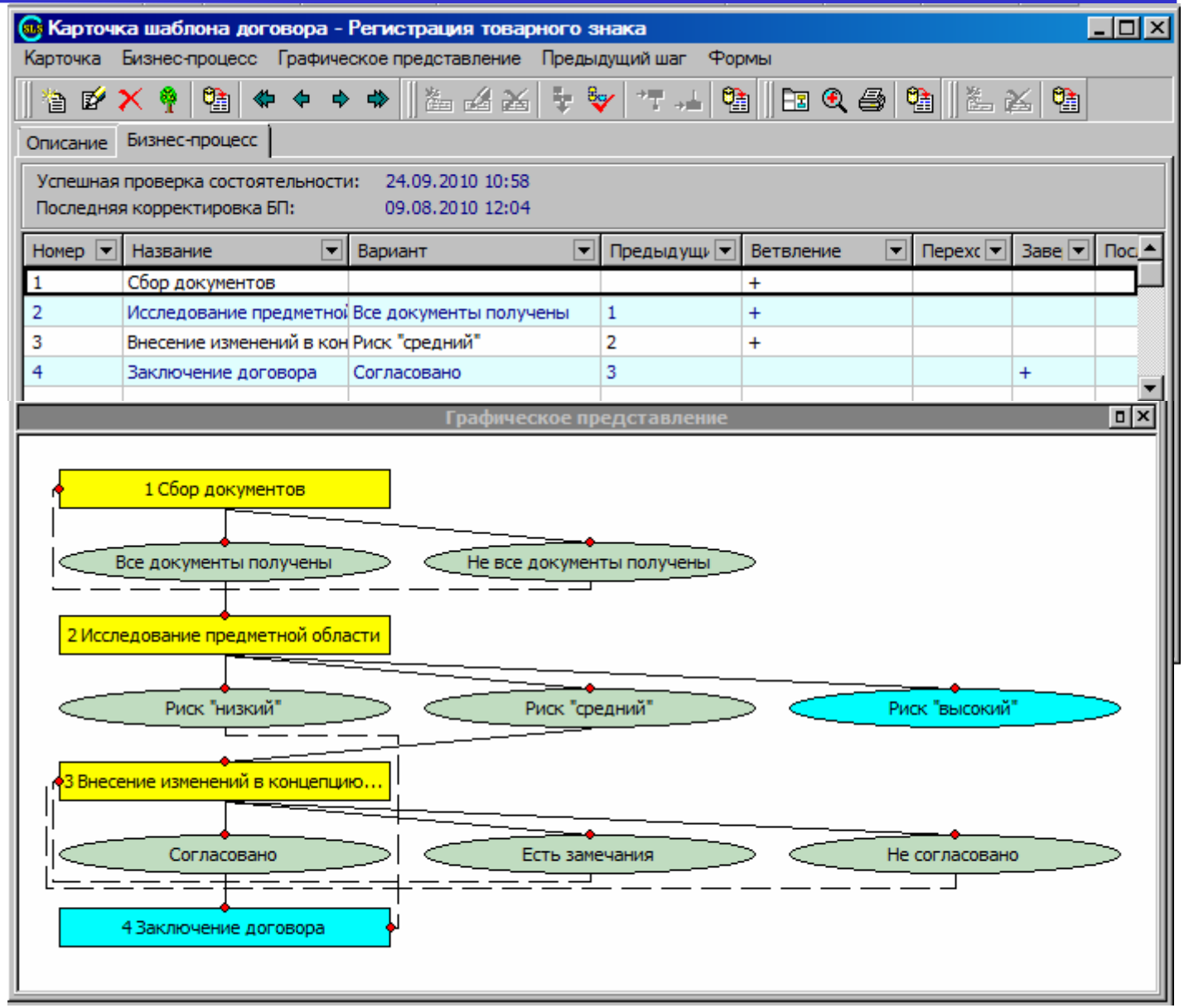
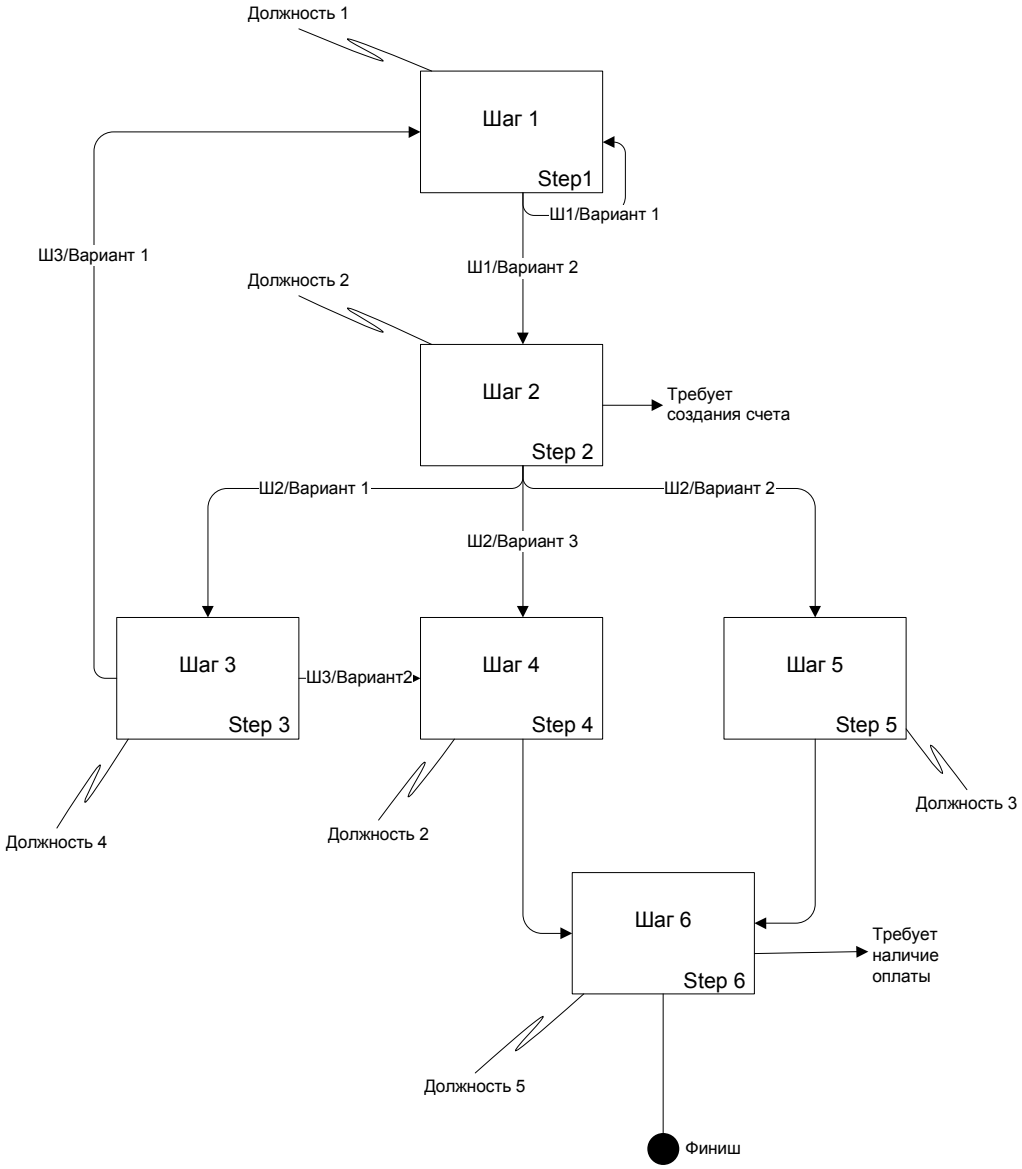
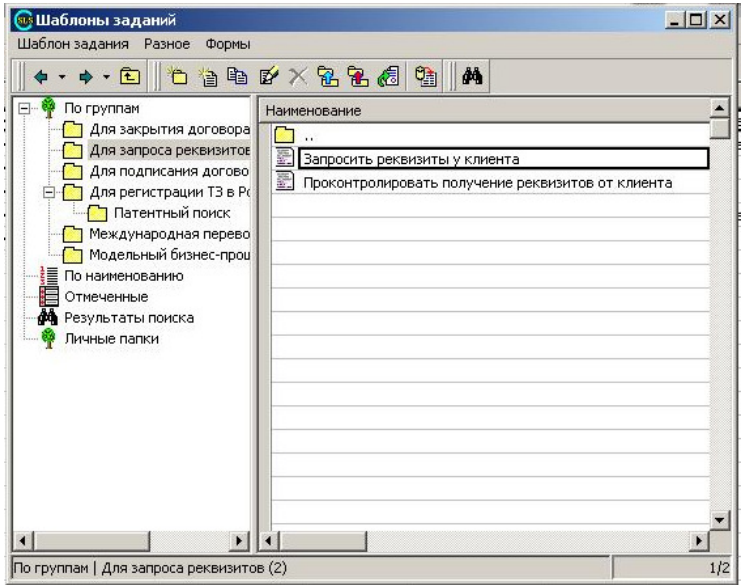


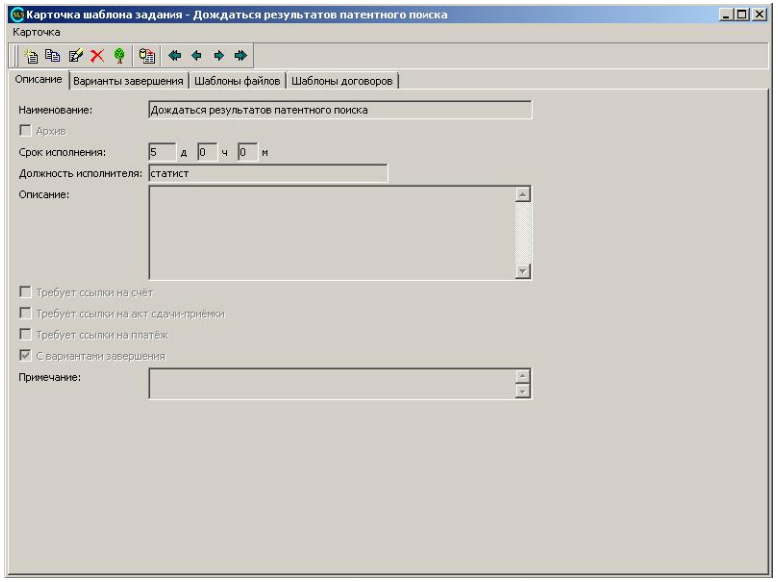
СХЕМА МОДЕЛЬНОГО БИЗНЕС-ПРОЦЕССА

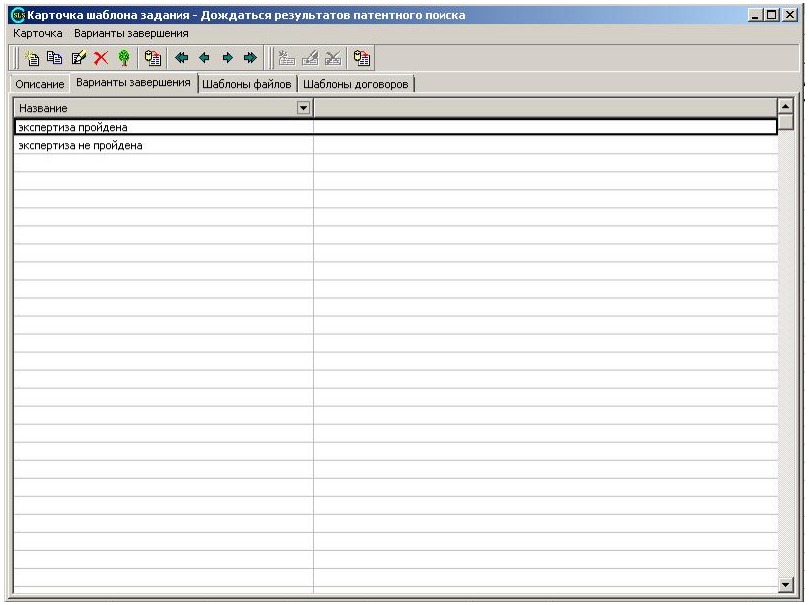


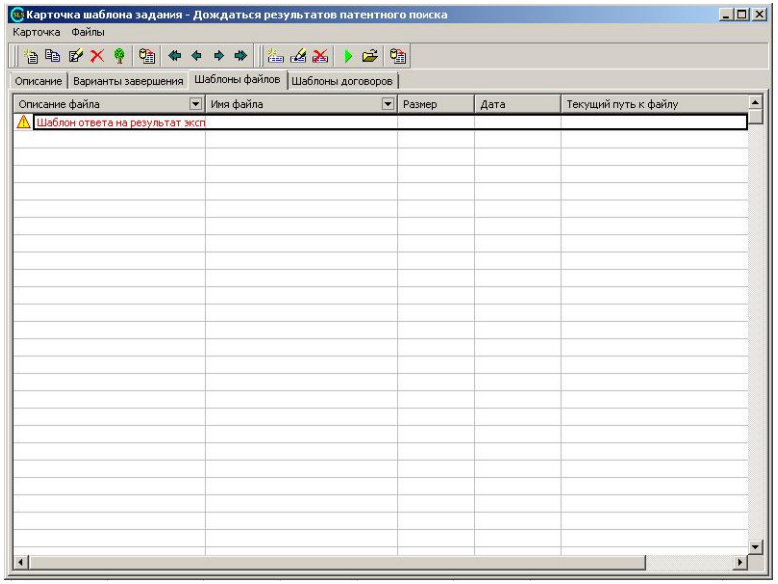
Из диаграммы видно, что модельный БП состоит из 6 шагов, в нем задействованы сотрудники на 5 разных должностях, между шагами используются различные варианты логических переходов. Некоторые из шагов требуют создания документов. Этот пример наглядно демонстрирует понятие логики бизнес-процесса, которая может быть задана в системе. Следует отметить, что каждый шаг состоит из набора заданий.

ЗАДАНИЯ – РЕАЛИЗАЦИЯ В СИСТЕМЕ









КОНТРОЛЬ ИСПОЛНЕНИЯ

Руководитель может посмотреть состояние исполнения заданий у подчиненных сотрудников, как по должностям, так и в общей совокупности по критериям «На исполнении», «Исполненные», «Просроченные».