Разработка элемента «бизнес-правило» для свободной системы RunaWFE Free в качестве производственной практики и ВКР

Зыкин Иван

ИАТЭ НИЯУ МИФИ

ООО «Процессные технологии»

RunaWFE Free

Цель работы

Расширение функциональности российской системы управления бизнес-процессами с открытым исходным кодом «RunaWFE Free»: реализация элемента «бизнес-правило».

Предметная область

BPM (*Business Process Management) – управление бизнес-процессами.*

BPMS (Business Process Management System) - система управления бизнес-процессами — технологическое программное обеспечение для поддержки концепции BPM.

RunaWFE Free BPMS

RunaWFE Free BPMS - это свободная система управления бизнес-процессами с открытым кодом.

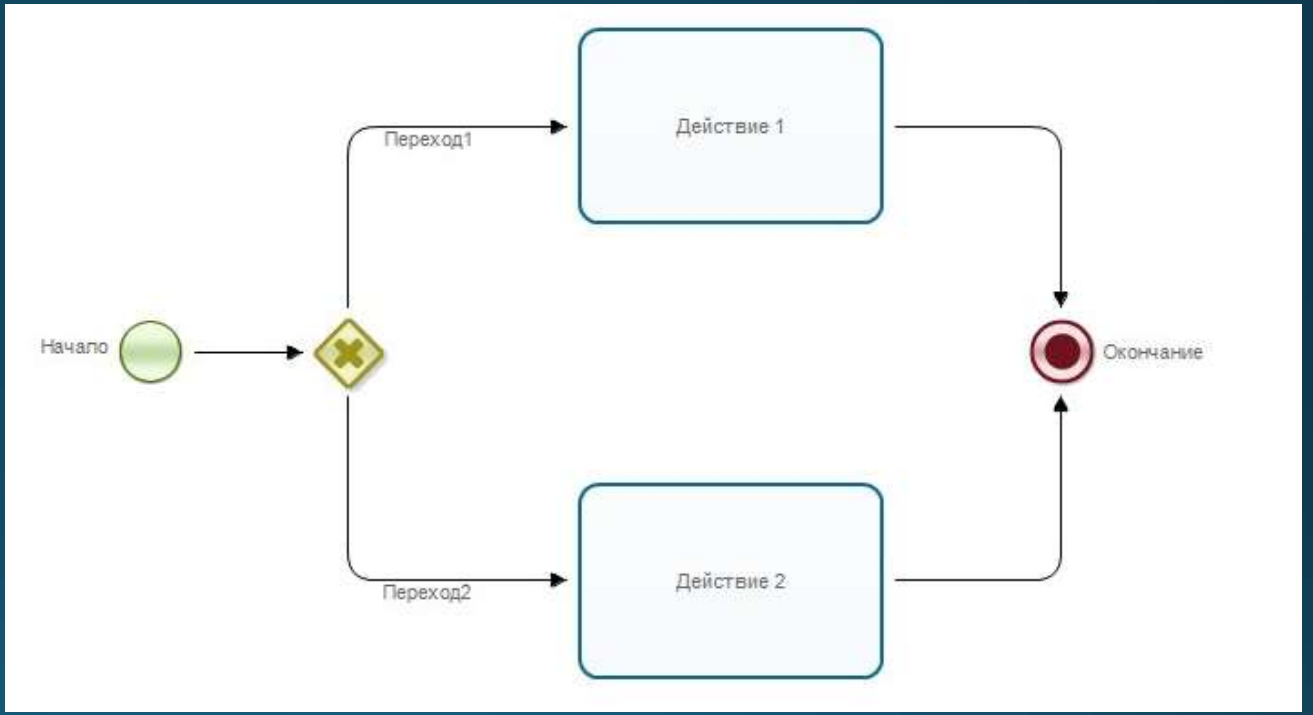
BPM система RunaWFE Free проста в установке и изучении, не требует специальных навыков в программировании.

Быстрый старт даёт возможность быстрого вхождения в систему, как аналитикам, использующим Графический дизайнер процессов для создания/изменения бизнес процессов в реальном времени, так и конечным пользователям, для которых упрощаются рутинные задачи и облегчается взаимодействие с другими участниками рабочего процесса.

Профессиональная версия системы RunaWFE Professional внесена в Единый реестр российских программ для электронных вычислительных машин и баз данных под номером 951 по классу ПО «системы управления процессами организации».

BPMN

Нотация моделирования бизнес-процессов (BPMN) — это метод составления блок-схем, отображающий этапы выполнения бизнес-процесса от начала до конца. BPMN-схемы наглядно и подробно демонстрируют последовательность рабочих действий и перемещение информационных потоков, необходимых для выполнения процесса, а потому являются одним из ключевых инструментов управления бизнесом.



Элемент «Бизнес-правило»

Функциональное предназначение элемента заключается в исполнении какой-либо формулы, при выполнении заданного условия.

Программные средства разработки

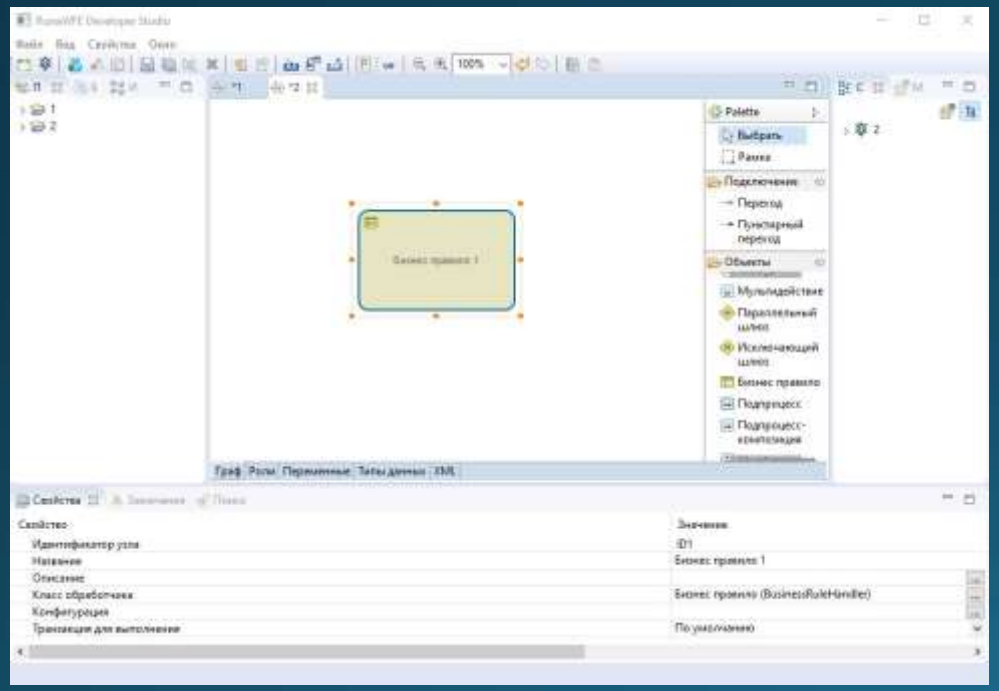
Разработка производилась на языке программирования Java с использованием фреймворка GEF (Graphical Editing Framework) в среде разработки Eclipse.

Этапы разработки

- BusinessRule
- BusinessRuleProvider
- BusinessRuleEditorDialog
- BusinessRuleModel
- BusinessRuleHandler

BusinessRule

Класс, непосредственно представляющий элемент(иконку) «Бизнес-правила» в среде разработки;



```java
package ru.runa.gpd.lang.model.bpmn;

import java.util.List;

public class BusinessRule extends Node implements Delegable {

    public BusinessRule() {
        setDelegationClassName(BusinessRuleHandler.class.getName());
    }

    @Override
    public String getDelegationType() {
        return HandlerArtifact.ACTION;
    }

    @Override
    public boolean testAttribute(Object target, String name, String value) {
        if (name.equals("delegableEditHandler")) {
            return false;
        }
        return super.testAttribute(target, name, value);
    }

    @Override
    protected boolean allowLeavingTransition(List<Transition> transitions) {
        return super.allowLeavingTransition(transitions) && transitions.size() == 0;
    }

}
```

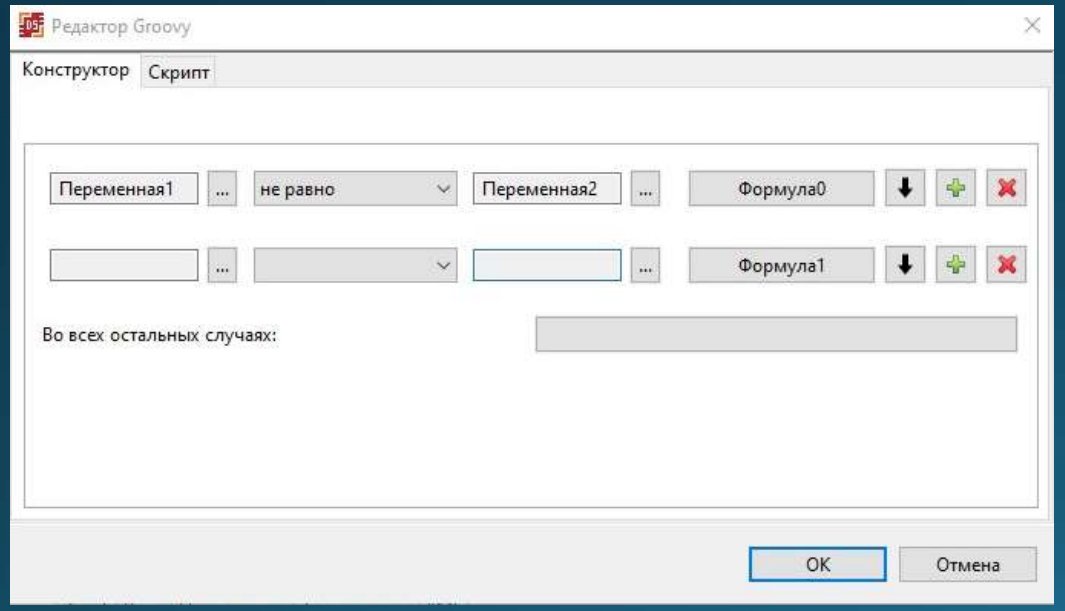
BusinessRuleProvider

BusinessRuleProvider инициализирует создание диалогового окна, при возникновении событий нажатия на элемент и производит валидацию конфигурации элемента.

```
public class BusinessRuleProvider extends DelegableProvider {
    @Override
    public String showConfigurationDialog(Delegable delegable) {
        ProcessDefinition definition = ((GraphElement) delegable).getProcessDefinition();
        BusinessRuleEditorDialog dialog = new BusinessRuleEditorDialog(definition, delegable.getDelegationConfiguration());
        if (dialog.open() == Window.OK) {
            return dialog.getResult();
        }
        return null;
    }

    @Override
    public boolean validateValue(Delegable delegable, List<ValidationError> errors) {
        String configuration = delegable.getDelegationConfiguration();
        if (configuration.trim().length() == 0) {
            errors.add(ValidationError.createLocalizedError((GraphElement) delegable, "delegable.invalidConfiguration.empty"));
        } else {
            Binding binding = new Binding();
            GroovyShell shell = new GroovyShell(binding);
            shell.parse(configuration);
        }
        return true;
    }

    @Override
    public List<String> getUsedVariableNames(Delegable delegable) throws Exception {
        List<Variable> variables = ((GraphElement) delegable).getProcessDefinition().getVariables(true, true);
        List<String> result = Lists.newArrayList();
        String configuration = "(" + delegable.getDelegationConfiguration() + ")";
        for (Variable variable : variables) {
            String variableName = String.format(VariableSearchVisitor.REGEX_SCRIPT_VARIABLE, variable.getScriptingName());
            if (Pattern.compile(variableName).matcher(configuration).find()) {
                result.add(variable.getName());
            }
        }
        return result;
    }

    @Override
    public String getConfigurationOnVariableRename(Delegable delegable, Variable currentVariable, Variable previewVariable) {
        return delegable.getDelegationConfiguration().replaceAll(Pattern.quote(currentVariable.getScriptingName()),
                Matcher.quoteReplacement(previewVariable.getScriptingName()));
    }

}
```

BusinessRuleEditorDialog

BusinessRuleEditorDialog – отвечающий за создание и отображение окна редактирования конфигурации разрабатываемого элемента.




```java
public BusinessRuleEditorDialog(ProcessDefinition definition, String initialValue) {
    super(definition, initialValue);
    if (this.initialValue.length() > 0) {
        try {
            initialModel = new BusinessRuleModel(initialValue, variables);
        } catch (Throwable e) {
            initialErrorMessage = e.getMessage();
            PluginLogger.logErrorWithoutDialog("", e);
        }
    }
}

@Override
protected void createConstructorView() {
    createExpressionLine();
    if (initialModel != null) {
        defaultFunction = initialModel.getDefaultFunction();
        for (int i = 1; i < initialModel.getIfExpressions().size(); i++) {
            createExpressionLine();
        }
    }
    createBottomComposite();
    ((ScrolledComposite) constructor.getParent()).setMinSize(constructor.computeSize(SWT.MIN, SWT.DEFAULT));
}
```

```
Button addLineButton = new Button(constructor, SWT.PUSH);
addLineButton.setImage(addImage);
addLineButton.addSelectionListener(new LoggingSelectionAdapter() {
    @Override
    protected void onSelection(SelectionEvent e) throws Exception {
        constructor.getChildren()[constructor.getChildren().length - 1].dispose();
        createExpressionLine();
        createBottomComposite();
        constructor.layout();
        ((ScrolledComposite) constructor.getParent()).setMinSize(constructor.computeSize(SWT.MIN, SWT.DEFAULT));
    }
});

Button deleteLineButton = new Button(constructor, SWT.PUSH);
deleteLineButton.setImage(deleteImage);
deleteLineButton.addSelectionListener(new LoggingSelectionAdapter() {
    @Override
    protected void onSelection(SelectionEvent e) throws Exception {
        expressionLines.remove(lineIndex);
        // expression line contains 6 elements
        for (int i = 0; i < 6; i++) {
            constructor.getChildren()[lineIndex * 6].dispose();
        }
        for (ExpressionLine expressionLine : expressionLines) {
            expressionLine.setLineIndex(expressionLines.indexOf(expressionLine));
        }
        // all lines removed
        if (constructor.getChildren().length == 1) {
            constructor.getChildren()[constructor.getChildren().length - 1].dispose();
            initialModel = null;
            createExpressionLine();
            createBottomComposite();
        }
        constructor.layout();
        ((ScrolledComposite) constructor.getParent()).setMinSize(constructor.computeSize(SWT.MIN, SWT.DEFAULT));
    }
});
```

```
private void createExpression(Composite parent) {
    Composite expressionComposite = new Composite(parent, SWT.NONE);
    GridData data = new GridData(GridData.FILL_HORIZONTAL);
    expressionComposite.setLayout(new GridLayout(3, true));
    data.horizontalSpan = 1;
    expressionComposite.setLayoutData(data);

    boolean isSimpleExpression = parent == constructor;
    int expressionIndex = (isSimpleExpression) ? SIMPLE_EXPRESSION_INDEX : variableBoxes.size();

    FilterBox[] expressionVariables = new FilterBox[2];
    expressionVariables[0] = new FilterBox(expressionComposite, VariableUtils.getVariableNamesForScripting(variables));
    expressionVariables[0].setData(DATA_INDEX_KEY, new int[] { expressionIndex, FIRST_VARIABLE_INDEX });
    expressionVariables[0].setData(DATA_EXPRESSION_LINE, this);
    expressionVariables[0].setSelectionListener(new FilterBoxSelectionHandler());
    expressionVariables[0].setLayoutData(getGridData());

    Combo operation = new Combo(expressionComposite, SWT.READ_ONLY);
    operation.setData(DATA_INDEX_KEY, new int[] { expressionIndex, OPERATION_INDEX });
    operation.setData(DATA_EXPRESSION_LINE, this);
    operation.addSelectionListener(new ComboSelectionHandler());
    operation.setLayoutData(getGridData());

    expressionVariables[1] = new FilterBox(expressionComposite, null);
    expressionVariables[1].setData(DATA_INDEX_KEY, new int[] { expressionIndex, SECOND_VARIABLE_INDEX });
    expressionVariables[1].setData(DATA_EXPRESSION_LINE, this);
    expressionVariables[1].setSelectionListener(new FilterBoxSelectionHandler());
    expressionVariables[1].setLayoutData(getGridData());
```

BusinessRuleModel

Класс BusinessRuleModel отвечает за хранение данных конфигурации и их представления в виде текста (скрипта).

```
public static class IfExpression {
    private List<Variable> firstVariables;
    private List<Object> secondVariables;
    private List<String> logicExpressions;
    private final List<Operation> operations;
    private List<int[]> brackets;
    private final String function;

    public IfExpression(String function, List<Variable> firstVariables, List<Object> secondVariables, List<Operation> operations,
            List<String> logicExpressions, List<int[]> brackets) {
        this.function = function;
        this.firstVariables = firstVariables;
        this.secondVariables = secondVariables;
        this.operations = operations;
        this.logicExpressions = logicExpressions;
        this.brackets = brackets;
    }

    public String generateCode() {
        StringBuffer buffer = new StringBuffer();
        buffer.append("if ( ");
        for (int i = 0; i < firstVariables.size(); i++) {

            for (int j = 0; j < brackets.get(i)[0]; j++) {
                buffer.append("(");
            }
            if (brackets.get(i)[0] != 0) {
                buffer.append(" ");
            }

            buffer.append(operations.get(i).generateCode(firstVariables.get(i), secondVariables.get(i)));

            if (brackets.get(i)[1] != 0) {
                buffer.append(" ");
            }
            for (int j = 0; j < brackets.get(i)[1]; j++) {
                buffer.append(")");
            }

            if (!logicExpressions.get(i).equals(BusinessRuleEditorDialog.NULL_LOGIC_EXPRESSION)) {
                if (logicExpressions.get(i).equals(BusinessRuleEditorDialog.OR_LOGIC_EXPRESSION)) {
                    buffer.append(" " + "||" + " ");
                }
                if (logicExpressions.get(i).equals(BusinessRuleEditorDialog.AND_LOGIC_EXPRESSION)) {
                    buffer.append(" " + "&&" + " ");
                }
            }
```

BusinessRuleHandler

Класс, отвечающий за обработку и выполнение скрипта.

```java
import ru.runa.wfe.execution.ExecutionContext;

public class BusinessRuleHandler extends ActionHandlerBase {

    @Override
    public void execute(ExecutionContext executionContext) throws Exception {
        GroovyDecisionHandler groovyDecisionHandler = new GroovyDecisionHandler();
        groovyDecisionHandler.setConfiguration(configuration);
        String function = groovyDecisionHandler.decide(executionContext);
        FormulaActionHandler formulaActionHandler = new FormulaActionHandler();
        formulaActionHandler.setConfiguration(function);
        formulaActionHandler.execute(executionContext);
    }
}
```

Интеграция элемента в систему

```
<element
      bpmn="businessRule"
      enabledInRegulationsByDefault="false"
      icon="businessRule.png"
      label="%label.element.businessRule"
      model="ru.runa.gpd.lang.model.bpmn.BusinessRule"
      type="node">
   <graphiti
         add="ru.runa.gpd.editor.graphiti.add.AddBusinessRuleNodeFeature"
         create="ru.runa.gpd.editor.graphiti.create.CreateElementFeature"
         doubleClick="ru.runa.gpd.editor.graphiti.update.DoubleClickDelegableFeature"
         fixedSize="false"
         height="10"
         layout="ru.runa.gpd.editor.graphiti.layout.LayoutStateNodeFeature"
         update="ru.runa.gpd.editor.graphiti.update.UpdateStateNodeFeature"
         width="15">
   </graphiti>
</element>
```

Заключение

Разработанный код был загружен на портал разработчиков свободного программного обеспечения GitHub в репозиторий компании "Процессные Технологии" - https://github.com/processtech

На основании работ по этому направлению автор прошел производственную практику в компании "Процессные Технологии" и защитили ВКР в ИАТЭ НИЯУ МИФИ.

Спасибо за внимание!