Оглавление

Перечень сокращений

ИМ – Имитационное Моделирование

СДС – Сложная Дискретная Система

IDE - Integrated Development Environment (Интегрированная Среда Разработки)

Терминология

Плагин – независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей

Парсинг – процесс анализа последовательности входных данных и преобразование их в необходимых формат

Парсер – компонент, выполняющий парсинг данных

Наблюдатель – поведенческий шаблон проектирования, создает механизм у класса, который позволяет получать экземпляру объекта этого класса оповещения от других объектов об изменении их состояния, тем самым наблюдая за ними

1. Введение

Имитационное моделирование (ИМ)[1] на ЭВМ находит широкое применение при исследовании и управлении сложными дискретными системами (СДС) и процессами, в них протекающими. К таким системам можно отнести экономические и производственные объекты, морские порты, аэропорты, комплексы перекачки нефти и газа, ирригационные системы, программное обеспечение сложных систем управления, вычислительные сети и многие другие. Широкое использование ИМ объясняется тем, что размерность решаемых задач и неформализуемость сложных систем не позволяют использовать строгие методы оптимизации. Эти классы задач определяются тем, что при их решении необходимо одновременно учитывать факторы неопределенности, динамическую взаимную обусловленность текущих решений и последующих событий, комплексную взаимозависимость между управляемыми переменными исследуемой системы, а часто и строго дискретную и четко определенную последовательность интервалов времени. Указанные особенности свойственны всем сложным системам.

Проведение имитационного эксперимента позволяет:

1. Сделать выводы о поведении СДС и ее особенностях:
    - без ее построения, если это проектируемая система
    - без вмешательства в ее функционирование, если это действующая система, проведение экспериментов над которой или слишком дорого, или небезопасно
    - без ее разрушения, если цель эксперимента состоит в определении пределов воздействия на систему
2. Синтезировать и исследовать стратегии управления.
3. Прогнозировать и планировать функционирование системы в будущем.
4. Обучать и тренировать управленческий персонал и т.д.

ИМ является эффективным, но и не лишенным недостатков, методом. Трудности использования ИМ, связаны с обеспечением адекватности описания системы, интерпретацией результатов, обеспечением стохастической сходимости процесса моделирования, решением проблемы размерности и т.п. К проблемам применения ИМ следует отнести также и большую трудоемкость данного метода.

Интеллектуальное ИМ, характеризующиеся возможностью использования методов искусственного интеллекта и прежде всего знаний, при принятии решений в процессе имитации, при управлении имитационным экспериментом, при реализации интерфейса пользователя, создании информационных банков ИМ, использовании нечетких данных, снимает часть проблем использования ИМ.

2. Предпроектное исследование

2.1. Основные положения языка РДО[2]

Основные положения системы РДО могут быть сформулированы следующим образом[1]:

- Все элементы СДС представлены как ресурсы, описываемые некоторыми параметрами. Ресурсы могут быть разбиты на несколько типов; каждый ресурс определенного типа описывается одними и теми же параметрами.

- Состояние ресурса определяется вектором значений всех его параметров; состояние СДС - значением всех параметров всех ресурсов.

- Процесс, протекающий в СДС, описывается как последовательность целенаправленных действий и нерегулярных событий, изменяющих определенным образом состояние ресурсов; действия ограничены во времени двумя событиями: событиями начала и событиями конца.

- Нерегулярные события описывают изменения состояния СДС, непредсказуемые в рамках продукционной модели системы (влияние внешних по отоношению к СДС факторов либо факторов, внутренних по отношению к ресурсам СДС). Моменты наступления нерегулярных событий случайны.

- Действия описываются операциями, которые представляют собой модифицированные продукционные правила, учитывающие временные связи. Операция описывает предусловия, которым должно удовлетворять состояние участвующих в операции ресурсов, и правила изменения состояния ресурсов в начале и в конце соответствующего действия.

- Множество ресурсов R и множество операций O образуют модель СДС.

2.2. Алгоритм поиска A*

Алгоритм A* - поиск по первому наилучшему совпадению на графе, который находит маршрут с наименьшей стоимостью от одной вершины к другой.

A* пошагово просматривает все пути, ведущие от начальной вершины в конечную, пока не найдет минимальный. Этот алгоритм раскрывает не все

вершины, а просматривает сначала те маршруты, которые «кажутся» ведущими к цели. Порядок обхода вершин определяется эвристической функцией «расстояние + стоимость».

2.3. Система имитационного моделирования RAO-studio[2]

Программный комплекс RAO-studio предназначен для разработки и отладки имитационных моделей на языке РДО. Основные цели данного комплекса - обеспечение пользователя легким в обращении, но достаточно мощным средством разработки текстов моделей на языке РДО, обладающим большинством функций по работе с текстами программ, характерных для сред программирования, а также средствами проведения и обработки результатов имитационных экспериментов.

В соответствии с основной целью программный комплекс решает следующие задачи:

- синтаксический разбор текста модели и настраиваемая подсветка синтаксических конструкций языка РДО
- открытие и сохранение моделей
- расширенные возможности для редактирования текстов моделей
- автоматическое завершение ключевых слов языка
- поиск и замена фрагментов текста внутри одного модуля модели
- поиск интересующего фрагмента текста по всей модели
- навигация по тексту моделей с помощью закладок
- наличие нескольких буферов обмена для хранения фрагментов текста
- вставка синтаксических конструкций языка и заготовок (шаблонов) для написания элементов модели
- настройка отображения текста моделей, в т.ч. скрытие фрагментов текста и масштабирование
- запуск и остановка процесса моделирования
- изменение режима моделирования
- изменение скорости работающей модели

- переключение между кадрами анимации в процессе моделирования
- отображение хода работы модели в режиме реального времени
- построение графиков изменения интересующих разработчика характеристик в режиме реального времени
- обработка синтаксических ошибок при запуске процесса моделирования
- обработка ошибок во время выполнения модели
- обеспечение пользователя справочной информацией

Программный комплекс состоит из двух частей:

- среды разработки (файл RAO-studio.exe под Windows и RAO-studio под Linux)
- файлов справок (rdo_lang_rus.qch - справка по языку РДО, rdo_studio_rus.qch - справка по программному комплексу, RAO-help.qhc - объединяет два последних для справочной системы)

2.4. Плагин Игра 5 для системы имитационного моделирования RAO-studio[2]

Плагин Игра 5 для системы имитационного моделирования представляет собой инструмент для проведения экспериментов с алгоритмом поиска A*.

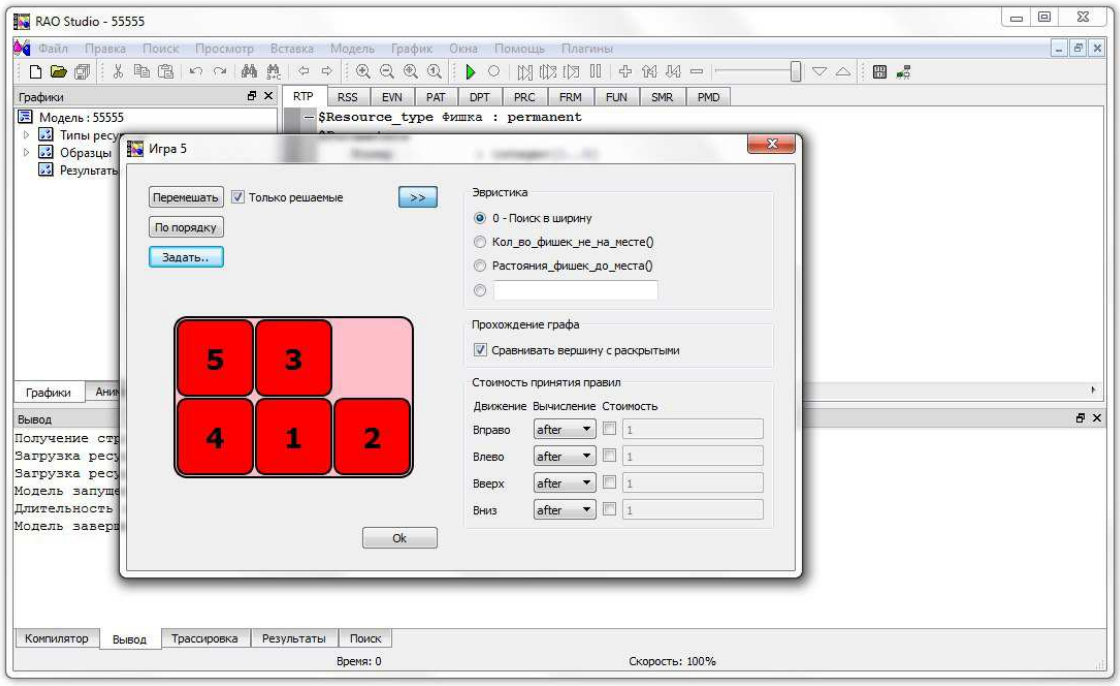


Рис. 1. Плагин Игра 5 для RAO-Studio

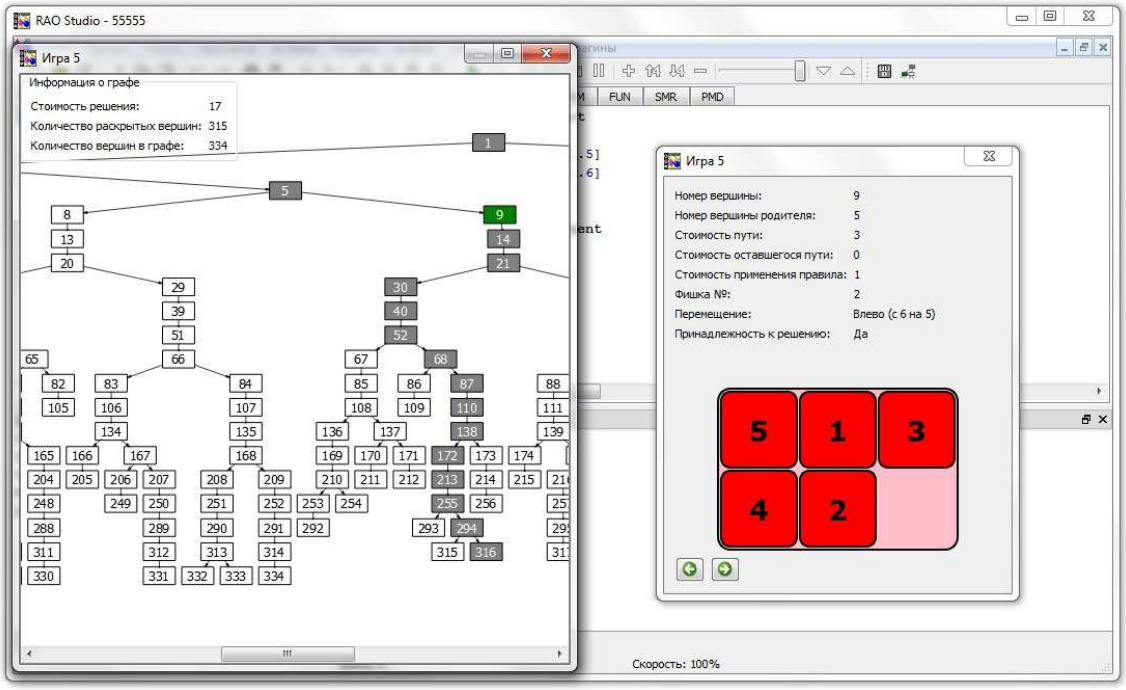


Рис. 2. Граф в плагине Игра 5 для RAO-Studio

2.5. Система имитационного моделирования Rao X

Система имитационного моделирования Rao X представляет собой плагин для интегрированной среды разработки Eclipse, позволяющий вести разработку имитационных моделей на языке РДО. Система написана на языке Java[3] и состоит из четырех основных компонентов:

- rao – компонент, производящий преобразование кода на языке РДО в код на языке Java
- rao.lib – библиотека системы. Этот компонент реализует ядро системы имитационного моделирования
- rao.ui – компонент, реализующий графический интерфейс системы с помощью библиотеки SWT[4]
- rao.tests – компонент, реализующий тестирование системы посредством Unit-тестов

3. Формирование ТЗ

3.1. Введение

Программный комплекс Rao X предназначен для разработки и отладки имитационных моделей на языке РДО. Основные цели данного комплекса - обеспечение пользователя легким в обращении, но достаточно мощным средством разработки текстов моделей на языке РДО, обладающим большинством функций по работе с текстами программ, характерных для сред программирования, а также средствами проведения и обработки результатов имитационных экспериментов.

Для проведения имитационных экспериментов на основе алгоритма A* необходимо иметь инструмент для задания исходных данных и анализа результатов, полученных после прогона модели.

3.2. Общие сведения

Основание для разработки: задание на курсовой проект.

Заказчик: Кафедра «Компьютерные системы автоматизации производства» МГТУ им. Н.Э. Баумана

Разработчик: студент кафедры «Компьютерные системы автоматизации производства» Зудина О.В.

Наименование темы разработки: «Разработка плагина Игра 5 для системы имитационного моделирования Rao X»

3.3. Назначение разработки

Разработать плагин Игра 5 для системы имитационного моделирования Rao X.

3.4. Требования к программе или программному изделию

3.4.1. Требования к функциональным характеристикам

Плагин Игра 5 должен удовлетворять следующим требованиям:

- Интеграция в Rao X
- Возможность задания начального положения различными способами
- Возможность переключения между эвристиками
- Автоматическая генерация текста модели
- Отображение текущего состояния поля при отображении информации о вершине графа

- Возможность задания различной стоимости правил
- Генерация случайным образом решаемых раскладок
- Изменение настроек алгоритма A*
- Создание проекта с помощью визарда

3.4.2. Требования к надежности

Основное требование к надежности направлено на поддержание в исправном и работоспособном состоянии ЭВМ, на которой происходит использование программного комплекса Rao X и плагина Игра 5.

3.4.3. Условия эксплуатации

- Эксплуатация должна производиться на оборудовании, отвечающем требованиями к составу и параметрам технических средств, и с применением программных средств, отвечающим требованиям к программной совместимости
- Аппаратные средства должны эксплуатироваться в помещениях с выделенной розеточной электросетью 220В ±10%, 50 Гц с защитным заземлением

3.4.4. Требования к составу и параметрам технических средств

Программный продукт должен работать на компьютерах со следующими характеристиками:

- объем ОЗУ не менее 2 Гб
- объем жесткого диска не менее 50 Гб
- микропроцессор с тактовой частотой не менее 2ГГц
- монитор с разрешением от 1366*768 и выше

3.4.5. Требования к информационной и программной совместимости

Система должна работать под управлением следующих ОС: Windows 7, Windows 8, Ubuntu 15.10.

3.4.6. Требования к маркировке и упаковке

Требования к маркировке и упаковке не предъявляются.

3.4.7. Требования к транспортированию и хранению

Требования к транспортированию и хранению не предъявляются.

3.5. Требования к программной документации

Требования к программной документации не предъявляются.

3.6. Стадии и этапы разработки

Плановый срок начала разработки – 20 июня 2015г.

Плановый срок окончания разработки – 26 сентября 2015г.

Этапы разработки:

- Концептуальный этап проектирования плагина
- Технический этап проектирования плагина
- Рабочий этап проектирования плагина

3.7. Порядок контроля и приемки

Контроль и приемка работоспособности системы осуществляются с помощью ручного тестирования плагина.

4. Концептуальный этап проектирования плагина

На концептуальном этапе проектирования требовалось:

- разработать концепцию зависимостей между плагином и системой имитационного моделирования Rao X
- подобрать формат для хранения конфигурации эксперимента
- разработать алгоритм автоматической генерации кода модели
- разработать интерфейс диалога, удобный для пользователя

4.1. Зависимости между плагином и системой имитационного моделирования Rao X

Плагин Игра 5 должен представлять собой библиотеку на языке Java, устанавливаемую в IDE Eclipse как подключаемый модуль. Для своей работы плагин должен требовать наличия установленной в IDE Eclipse системы Rao X. В свою очередь, система Rao X не должна зависеть от плагина Игра 5.

4.2. Хранение конфигурации эксперимента

Конфигурации, необходимые для хранения:

- положения фишек
- название эвристики
- код эвристики
- решаемость ситуации
- сравнение вершин графа
- стоимости правил
- вычисление стоимости пути

Для хранения данных конфигураций следует использовать следующие структуры:

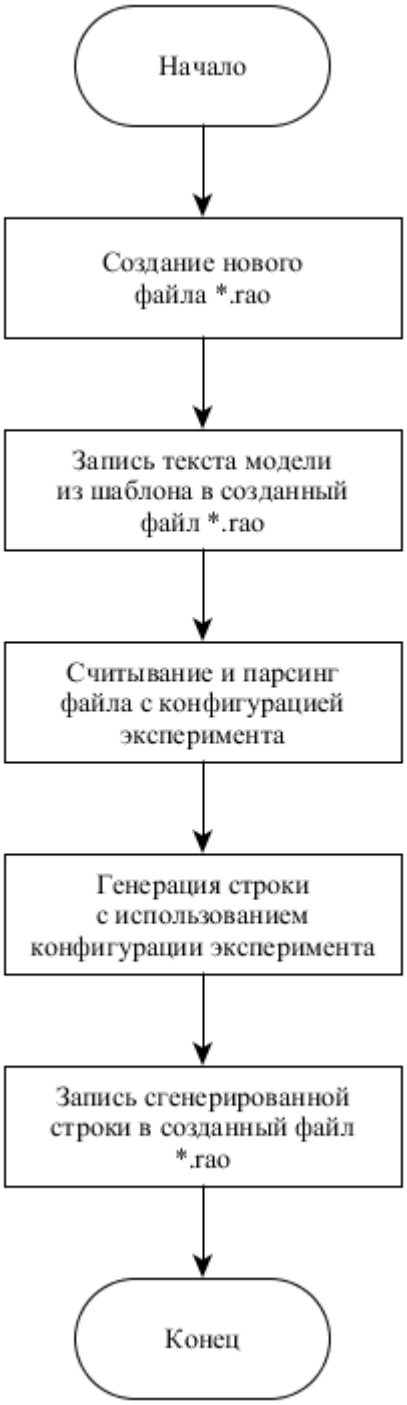
- упорядоченный список (положения фишек)
- пара «имя-значение» (название эвристики, код эвристики и т.д.)

К формату хранения предъявляются следующие требования:

- возможность парсинга данных
- удобство чтения для человека

4.3. Алгоритм автоматической генерации кода модели

Генерация кода модели осуществляется на основе двух файлов: шаблона модели в формате rao и файла конфигурации эксперимента в формате JSON[4]. В результате должен быть получен файл модели в формате rao.



4.4. Интерфейс диалога

Диалог плагина Игра 5 должен обладать следующим функционалом:

- отображение поля

- задание исходного положения фишек различными способами:
    - перестановка фишек с помощью курсора
    - генерация случайным образом
    - ввод с клавиатуры
    - расстановка фишек по порядку
- выбор эвристики
- порядок вычисления стоимости пути
- стоимости правил
- сравнение вершин графа
- запуск прогона с последующим открытием графа

5. Технический этап проектирования плагина

5.1. Выбор формата хранения конфигурации эксперимента

Для хранения конфигурации эксперимента был выбран формат *JSON*[4] – текстовый формат обмена данными, легко читаемый человеком и имеющий возможность парсинга.

JSON[4] использует две структуры: пара *«имя-значение»* и упорядоченный список. Элементами структуры могут быть объекты различных типов: строки, числа, массивы и т.п.

Для чтения и парсинга конфигурации целесообразно использование библиотеки *JSON.simple*, разработчиком которой является *Google Inc.*

5.2. Проектирование диалога

5.2.1. Выбор компонентов диалога

Для реализации функционала плагина были использованы компоненты, легко встраиваемые в среду IDE Eclipse.

- Стандартные SWT-компоненты:
  - *Button* (генерация случайным образом положения фишек, задание порядка фишек, запуск прогона)
  - *Radio Button* (задание параметра решаемости положения фишек)
  - *Check Button* (использование нестандартной стоимости правил)
  - *Text* (задание положения фишек с клавиатуры, задание стоимости правил)
  - *Combo* (выбор эвристики по названию)
- Стандартные Xtext-компоненты:
  - *Embedded Editor* (ввод кода эвристики)
- Собственная реализация расширений над SWT-компонентами:
  - *Tile Button* (отображение состояния поля, возможность задания положения фишек с помощью курсора)

5.2.2. Отображение диалога

Диалог должен быть расположен в области редактора модели среды разработки IDE Eclipse, занимать всю область, быть масштабируемым с возможностью прокрутки и отображения скрытых частей диалога.

Открытие диалога реализуется посредством двойного клика по файлу конфигурации, который содержится в проекте и имеет имя *config.json*.

5.3. Визард создания проектов

Для запуска плагина был спроектирован визард, работающий по аналогии со стандартными визардами, предоставляемыми средой разработки IDE Eclipse.

Визард представляет собой диалоговое окно класса *Game5Wizard*, в котором содержится страница класса *Game5WizardPage* с интерфейсом для ввода имени проекта.

5.4. Отображение состояния поля в окне графа

При анализе результатов эксперимента необходимо наглядное отображение состояния игрового поля в текущей вершине графа.

Поле состоит из определенного количества элементов *Tile Button*, расположенных в соответствующем порядке с фишками. Реализуется посредством класса *GraphManager*, с помощью подписчиков класса *Subscriber*.

6. Рабочий этап проектирования плагина

На рабочем этапе проектирования системы были реализованы разработанные на предыдущих этапах схемы и концепции.

6.1. Хранение конфигурации эксперимента

6.1.1. Запись в файл конфигурации

При изменении конфигурации эксперимента происходит запись в файл:

```
object.put(key, value);
```

Key – "ключ" для конфигурации.
Value – значение конфигурации.

Примеры "ключей": *places*, *solvable*, *heuristic*, *enableUp*.
Примеры значений: "*1*, *2*, *3*, *4*, *5*, *6*", "*all*", "*Поиск_в_ширину()*", "*true*".

6.2.Чтение и парсинг конфигурации из файла

Для чтения файла конфигурации используется объект класса *JSONParser:*

```
final JSONParser parser = new JSONParser();
JSONObject object = (JSONObject) parser.parse(new InputStreamReader(
      new FileInputStream(ResourcesPlugin.getWorkspace()
            .getRoot().getLocation()
            .append(configIFile.getFullPath()).toString()),
      StandardCharsets.UTF_8));
```

С его помощью создается объект класса *JSONObject*, из которого считываются данные:

```
object.get(key);
```

Пример парсинга конфигурации:

```
final JSONArray places = (JSONArray) object.get("places");
```

6.3.Реализация диалога плагина

6.3.1. Реализация отображения диалога

Диалог плагина описан в классе *Game5View*.

Реализация масштабирования и заполнения области:

```
ScrolledComposite scrolledComposite = new ScrolledComposite(parent,
                        SWT.H_SCROLL | SWT.V_SCROLL | SWT.FILL);
scrolledComposite.setExpandHorizontal(true);
scrolledComposite.setExpandVertical(true);
scrolledComposite.setLayout(new FillLayout());
```

Реализация открытия диалога:

```
final IWorkbenchPage page = PlatformUI.getWorkbench()
      .getActiveWorkbenchWindow().getActivePage();
page.openEditor(new FileEditorInput(configIFile), Game5View.ID);
```

6.3.2. Реализация компонентов

Стандартные компоненты были реализованы в соответствии с документацией библиотеки.

Пример реализации SWT-компонента:

```
final Group traverseGraph = new Group(composite, SWT.SHADOW_IN);
traverseGraph.setText("Traverse graph:");
traverseGraph.setBackground(color);
traverseGraph.setLayout(gridLayout);
final Button compareTops = new Button(traverseGraph, SWT.CHECK);
compareTops.setText("Compare tops");
compareTops.setSelection((boolean) object.get("compare"));
```

Пример реализации Xtext-компонента:

```
final Injector injector = ru.bmstu.rk9.rao.ui.RaoActivatorExtension
      .getInstance()
      .getInjector(
            ru.bmstu.rk9.rao.ui.RaoActivatorExtension.RU_BMSTU_RK9_RAO_RAO);
final EmbeddedEditorFactory factory = injector
      .getInstance(EmbeddedEditorFactory.class);
final EditedResourceProvider resourceProvider = injector
      .getInstance(EditedResourceProvider.class);
final EmbeddedEditor embeddedEditor = factory
      .newEditor(resourceProvider).showErrorAndWarningAnnotations()
      .withParent(editorGroup);
editor = embeddedEditor.createPartialEditor("", object.get("code")
      .toString(), "", false);
IXtextDocument document = embeddedEditor.getDocument();
```

Компонент *Tile Button* представляет собой расширение SWT-компонента. Пример реализации:

```
new TileButton(boardGroup, SWT.NONE, String
      .valueOf(places.indexOf(String.valueOf(i + 1)) + 1), i + 1)
```

6.3.3. Реализация функционала компонентов

Для реализации функционала использовались стандартные наблюдатели библиотеки SWT и расширения над ними.

Пример реализации стандартного наблюдателя:

```
compareTops.addSelectionListener(new SelectionListener() {
      @Override
      public void widgetSelected(SelectionEvent arg0) {
            object.put("compare", compareTops.getSelection());
            setDirty(true);
      }
```

```
                @Override
                public void widgetDefaultSelected(SelectionEvent arg0) {
                }
        });
```

Было создано три различных класса расширений стандартных наблюдателей:

- `ConfigurationListener implements SelectionListener`
- `CostButtonListener implements SelectionListener`
- `ConfigurationKeyListener implements KeyListener`

Пример реализации расширения:

```
heuristicList.addSelectionListener(new ConfigurationListener(
                        "heuristic", () -> heuristicList.getText()));
```

6.4.Элементы визарда

На странице визарда класса *Game5WizardPage* для реализации интерфейса были использованы стандартные SWT-компоненты.

Особенностью визарда является проверка на корректность введенного имени:

```
String projectName = projectNameText.getText();

      if (projectName.isEmpty()) {
            setDescription("Enter a project name");
            setPageComplete(false);
            return;
      }
      if (!isValidJavaIdentifier(projectName)) {
            setDescription("Project name is not a valid Java identifier.");
            setPageComplete(false);
            return;
            }
      if (isJavaKeyword(projectName)) {
            setDescription("Project name can not be a Java keyword.");
            setPageComplete(false);
            return;
            }
      if (projectName.equals("model")) {
            setDescription("\"model\" is an invalid name for Rao project");
            setPageComplete(false);
            return;
            }
      if (root.getProject(projectName).exists()) {
            setDescription("A project with this name already exists.");
            setPageComplete(false);
            return;
            }

      setDescription("Create a Rao Game5 project in the workspace.");
      setPageComplete(true);
```

7. Апробирование разработанного плагина

Апробирование разработанного плагина осуществлялось при помощи многократного тестирования функционала. Выявленные в процессе тестирования ошибки и недочеты были исправлены на этапе рабочего проектирования.

После окончания этапа рабочего проектирования разработанный плагин использовался в качестве ПО для лабораторной работы по курсу Моделирование дискретных производственных процессов. В ходе проведения лабораторной работы серьезных ошибок и недочетов выявлено не было.

8. Заключение

По завершению работы над данным курсовым проектом были получены следующие результаты:

- Реализован механизм автоматической генерации кода модели с помощью шаблона и файла конфигурации эксперимента.
- Реализованы зависимости между системой имитационного моделирования Rao X и плагином Игра 5.
- Реализован алгоритм формирования положения фишек случайным образом при заданном ограничении решаемости.
- Разработан визард создания проектов.
- Разработан плагин Игра 5 для системы имитационного моделирования Rao X.
- Проведена лабораторная работа по курсу Моделирование дискретных производственных процессов с использованием данного плагина в качестве ПО.

Список используемых источников

1. **Емельянов В.В., Ясиновский С.И.** Введение в интеллектуальное имитационное моделирование сложных дискретных систем и процессов. Язык РДО. - М.: "Анвик", 1998. - 427 с., ил. 136.
2. **Документация по языку РДО** [http://raox.ru/docs/reference/base_types_and_functions.html]
3. **Java™ Platform, Standard Edition 7. API Specification.** [http://docs.oracle.com/javase/7/docs/api/]
4. **JSON.simple Documentation** [https://code.google.com/p/json-simple/]
5. **SWT Documentation** [https://www.eclipse.org/swt/docs.php]
6. **Map Documentation** [ps://docs.oracle.com/javase/8/docs/api]

Список использованного программного обеспечения

1. RAO-Studio v2.4.3
2. Eclipse IDE for Java Developers Luna Service Release 1 (4.4.1)
3. openjdk version "1.8.0_40-internal"
4. UMLet v13.3
5. yEd Graph Editor v3.14.4
6. Inkscape v0.48.4
7. Microsoft® Office Word 2010
8. Microsoft® Office Excel 2010
9. Microsoft® Visio 2013

Приложение 1 – Шаблон текста модели

```
enum Место_дырки {справа, слева, сверху, снизу, дырки_рядом_нет}

type Фишка {
    int Номер;
    int Местоположение;
}

type Дырка_t {
    int Место;
}

rule Перемещение_фишки (Место_дырки Куда_перемещать, int На_сколько_перемещать) {
      relevant _Фишка =  Фишка.select(Где_дырка(_Фишка.Местоположение) ==
Куда_перемещать).first();
      relevant _Дырка = Дырка_t.select(any).first();
      set execute() {
            _Фишка.Местоположение = _Фишка.Местоположение + На_сколько_перемещать;
            _Дырка.Место = _Дырка.Место - На_сколько_перемещать;
      }
}

constant int Длина_поля = 3;

int Ряд(int Местоположение) {
      return (Местоположение - 1)/Длина_поля + 1;
}

int Остаток_от_деления(int Делимое, int Делитель) {
      int Целая_часть  = Делимое/Делитель;
      int Макс_делимое = Делитель * Целая_часть;
      return Делимое -  Макс_делимое;
}

int Столбец(int Местоположение) {
      return Остаток_от_деления(Местоположение - 1,Длина_поля) + 1;
}

Место_дырки Где_дырка(int _Место) {
      if (Столбец(_Место) == Столбец(Дырка.Место) and Ряд(_Место) ==
Ряд(Дырка.Место)+ 1) return Место_дырки.сверху;
      if (Столбец(_Место) == Столбец(Дырка.Место) and Ряд(_Место) ==
Ряд(Дырка.Место)- 1) return Место_дырки.снизу;
      if (Ряд(_Место) == Ряд(Дырка.Место) and Столбец(_Место) ==
Столбец(Дырка.Место)- 1) return Место_дырки.справа;
      if (Ряд(_Место) == Ряд(Дырка.Место) and Столбец(_Место) ==
Столбец(Дырка.Место)+ 1) return Место_дырки.слева;
      return Место_дырки.дырки_рядом_нет;
}

int Поиск_в_ширину() {
      return 0;
}

int Фишка_на_месте(int _Номер, int _Место) {
      if (_Номер == _Место) return 1;
      else                  return 0;
}

int Кол_во_фишек_не_на_месте() {
      return 5 - (Фишка_на_месте(Фишка1.Номер, Фишка1.Местоположение)+
```

```
                    Фишка_на_месте(Фишка2.Номер, Фишка2.Местоположение)+
                    Фишка_на_месте(Фишка3.Номер, Фишка3.Местоположение)+
                    Фишка_на_месте(Фишка4.Номер, Фишка4.Местоположение)+
                    Фишка_на_месте(Фишка5.Номер, Фишка5.Местоположение));
}

int Расстояние_фишки_до_места(int Откуда, int Куда) {
      return Math.abs(Ряд(Откуда)-Ряд(Куда)) + Math.abs(Столбец(Откуда)-
Столбец(Куда));
}

int Расстояния_фишек_до_мест() {
      return Расстояние_фишки_до_места(Фишка1.Номер, Фишка1.Местоположение)+
             Расстояние_фишки_до_места(Фишка2.Номер, Фишка2.Местоположение)+
             Расстояние_фишки_до_места(Фишка3.Номер, Фишка3.Местоположение)+
             Расстояние_фишки_до_места(Фишка4.Номер, Фишка4.Местоположение)+
             Расстояние_фишки_до_места(Фишка5.Номер, Фишка5.Местоположение);
}
```

Приложение 2 – Начальная конфигурация эксперимента

```
{       "order":["1","2","3","4","5","6"],
        "places":["1","2","3","4","5","6"],
        "solvable":"all",
        "compare":true,
        "heuristic":"Поиск_в_ширину()",
        "computeLeft":"After",
        "computeRight":"After",
        "computeUp":"After",
        "computeDown":"After",
        "costLeft":1,
        "costRight":1,
        "costDown":1,
        "costUp":1,
        "enableLeft":false,
        "enableRight":false,
        "enableUp":false,
        "enableDown":false,
        "code":"",
        }
```

Приложение 3 – Автоматически сгенерированный код модели на основе шаблона

```
enum Место_дырки {справа, слева, сверху, снизу, дырки_рядом_нет}

type Фишка {
    int Номер;
    int Местоположение;
}

type Дырка_t {
    int Место;
}

rule Перемещение_фишки (Место_дырки Куда_перемещать, int На_сколько_перемещать) {
      relevant _Фишка =  Фишка.select(Где_дырка(_Фишка.Местоположение) ==
Куда_перемещать).first();
      relevant _Дырка = Дырка_t.select(any).first();
      set execute() {
            _Фишка.Местоположение = _Фишка.Местоположение + На_сколько_перемещать;
            _Дырка.Место = _Дырка.Место - На_сколько_перемещать;
      }
}

constant int Длина_поля = 3;

int Ряд(int Местоположение) {
      return (Местоположение - 1)/Длина_поля + 1;
}

int Остаток_от_деления(int Делимое, int Делитель) {
      int Целая_часть  = Делимое/Делитель;
      int Макс_делимое = Делитель * Целая_часть;
      return Делимое -  Макс_делимое;
}

int Столбец(int Местоположение) {
      return Остаток_от_деления(Местоположение - 1,Длина_поля) + 1;
}

Место_дырки Где_дырка(int _Место) {
      if (Столбец(_Место) == Столбец(Дырка.Место) and Ряд(_Место) ==
Ряд(Дырка.Место)+ 1) return Место_дырки.сверху;
      if (Столбец(_Место) == Столбец(Дырка.Место) and Ряд(_Место) ==
Ряд(Дырка.Место)- 1) return Место_дырки.снизу;
      if (Ряд(_Место) == Ряд(Дырка.Место) and Столбец(_Место) ==
Столбец(Дырка.Место)- 1) return Место_дырки.справа;
      if (Ряд(_Место) == Ряд(Дырка.Место) and Столбец(_Место) ==
Столбец(Дырка.Место)+ 1) return Место_дырки.слева;
      return Место_дырки.дырки_рядом_нет;
}

int Поиск_в_ширину() {
      return 0;
}

int Фишка_на_месте(int _Номер, int _Место) {
      if (_Номер == _Место) return 1;
      else                  return 0;
}

int Кол_во_фишек_не_на_месте() {
```

```
	return 5 - (Фишка_на_месте(Фишка1.Номер, Фишка1.Местоположение)+
		    Фишка_на_месте(Фишка2.Номер, Фишка2.Местоположение)+
		    Фишка_на_месте(Фишка3.Номер, Фишка3.Местоположение)+
		    Фишка_на_месте(Фишка4.Номер, Фишка4.Местоположение)+
		    Фишка_на_месте(Фишка5.Номер, Фишка5.Местоположение));
}

int Расстояние_фишки_до_места(int Откуда, int Куда) {
	return Math.abs(Ряд(Откуда)-Ряд(Куда)) + Math.abs(Столбец(Откуда)-
Столбец(Куда));
}

int Расстояния_фишек_до_мест() {
	return Расстояние_фишки_до_места(Фишка1.Номер, Фишка1.Местоположение)+
		Расстояние_фишки_до_места(Фишка2.Номер, Фишка2.Местоположение)+
		Расстояние_фишки_до_места(Фишка3.Номер, Фишка3.Местоположение)+
		Расстояние_фишки_до_места(Фишка4.Номер, Фишка4.Местоположение)+
		Расстояние_фишки_до_места(Фишка5.Номер, Фишка5.Местоположение);
}

resource Фишка1 = Фишка.create(1, 1);
resource Фишка2 = Фишка.create(2, 2);
resource Фишка3 = Фишка.create(3, 3);
resource Фишка4 = Фишка.create(4, 4);
resource Фишка5 = Фишка.create(5, 5);
resource Дырка = Дырка_t.create(6);

search Расстановка_фишек {
	set init() {
		setCondition(exist(Фишка: Фишка.Номер != Фишка.Местоположение));
		setTerminateCondition(forAll(Фишка: Фишка.Номер ==
Фишка.Местоположение));
		compareTops(true);
		evaluateBy(Поиск_в_ширину());
	}
	activity Перемещение_вправо checks Перемещение_фишки(Место_дырки.справа,
1).setValueAfter(1);
	activity Перемещение_влево checks Перемещение_фишки(Место_дырки.слева,  -
1).setValueAfter(1);
	activity Перемещение_вверх checks Перемещение_фишки(Место_дырки.сверху,  -
3).setValueAfter(1);
	activity Перемещение_вниз checks Перемещение_фишки(Место_дырки.снизу,
3).setValueAfter(1);
}
```