Выпускная квалификационная работа бакалавра по теме

Разработка программного модуля визуализации диаграмм процессов по спецификации на языке Reflex

Выполнила **Беленькая София Евгеньевна**

Научный руководитель: **Зюбин В.Е.,** зав. кафедрой КТ, д. т. н., доцент, зав. лаб. ИАиЭ СО РАН.

Соруководитель: **Розов А. С.**, старший преподаватель кафедры КТ.

Построение диаграмм

Итеративная модель разработки ПО

Построение диаграмм по коду необходимо:

- при работе с итеративной моделью для создания текущей рабочей документации;
- при создании выходной документации.

Языки описания управляющих алгоритмов

- языки процесс-ориентированного программирования (ПОП);
  - Reflex
  - Industrial C
- языки стандартов МЭК 61131-3.

При работе с ПОП диаграммы процессов рисуются вручную, что приводит к ряду проблем:

- занимает значительное время;
- может быть причиной ошибок.

Цель работы и задачи

Цель работы: разработка программного модуля визуализации диаграмм процессов по спецификации на языке Reflex.

Задачи:

- провести анализ:
  - специфики ПОП на языке Reflex;
  - диаграмм, использующихся для анализа кода;
  - средств визуализации диаграмм для языков общего назначения;
  - средств визуализации графов.
- спроектировать систему:
  - сформулировать требования к создаваемому программному модулю;
  - разработать диаграммы для отображения связей процессов;
  - определить формат представления диаграмм;
- разработать архитектуру модуля, реализовать модуль визуализации;
- провести тестирование созданной реализации, опробовать ее на практике;

Специфика языка Reflex

```
ПРОЦ Разогрев{ /* готовит, если дверца закрыта и время ненулевое,ес
ИЗ ПРОЦ Инициализация К_ДВЕРЦА, У_ЗВОНОК,У_РАЗОГРЕВ, ВремяГотовки;
    СОСТ Начало
    {
        ЕСЛИ (К_ДВЕРЦА = ОТКР)
        {
            ВремяГотовки = 0;
        } ИНАЧЕ
        {
            ЕСЛИ (ВремяГотовки != 0)
            {
                У_РАЗОГРЕВ = ВКЛ;
                В СЛЕДУЮЩЕЕ;
            }
        }
    }

    СОСТ Разогрев
    {
        ЕСЛИ (К_ДВЕРЦА != ОТКР)
        {
            У_РАЗОГРЕВ = ВЫКЛ;
            В СОСТ ОжиданиеЗакрытияДверцы;
        }
        ТАЙМАУТ ВремяГотовки
        {
            У_РАЗОГРЕВ = ВЫКЛ;
            В СЛЕДУЮЩЕЕ;
        }
    }
```

- Программа состоит из описания процессов.
- Процессы представлены автоматами состояний.
- Исполнение происходит в кооперативной модели многопоточности.
- Процессы взаимодействуют по данным и по управлению.

Сравнительный анализ средств визуализации диаграмм для языков общего назначения

Название	Реверсивный инжиниринг	Кодогенерация	Возможность модификации	Динамическое построение	Возможность скрывать компоненты классов	Количество диаграмм
Class Designer для Visual Studio	+	-	+	+	+	1?
Star UML	-	-	+			11
Astah UML	+	+	+			9
MagicDraw	+	+	+	+ *		24
Software Ideas Modeler	+	+	+	+?	+	14

Продолжение таблицы.

BOUML	+	+	+	-?	+	10+
Visual paradigm	+	+	+			28
Rational Rose	+	+	+	+	+?	5+
Enterprise Architect.	+	+	+			12
IntelliJ Idea	+	-	+	+?	+	3
Sybase PowerDesigner	+	+	+	-?	+	3+
NetBeans	+	-	+	-?	-?	5+
Lab view	+	+	+		-	1
NClass	-	+	+			1
Altova UModel 2008	+	+	+			14+

Требования

- Возможности редактирования:
  - Перетаскивание блоков с помощью мыши (drag-and-drop).
  - Возможность изменять имена компонентов диаграмм.
  - Возможность удаления компонента диаграммы.
  - Возможность скрыть или показать подписи над стрелками в диаграммах.
- Визуализация диаграмм:
  - Автоматическая укладка диаграмм на плоскость.
  - Автоматическое разделение несвязанных областей графов на разные диаграммы.
  - Возможность выбирать процессы для визуализации их взаимодействия.

Требования (продолжение)

- Архитектура модуля:
  - Наличие графического интерфейса, взаимодействующего с модулем визуализации через API.
  - Обновление диаграммы по нажатию кнопки (по явному вызову).
- Исходя из анализа ПОП, необходима визуализация следующих диаграмм:
  - Диаграмм состояний процесса.
  - Диаграмм связи процессов по данным.
  - Диаграмм связи процессов по управлению.
- Возможность сохранения диаграммы в отдельный файл.

Диаграмма состояний процесса

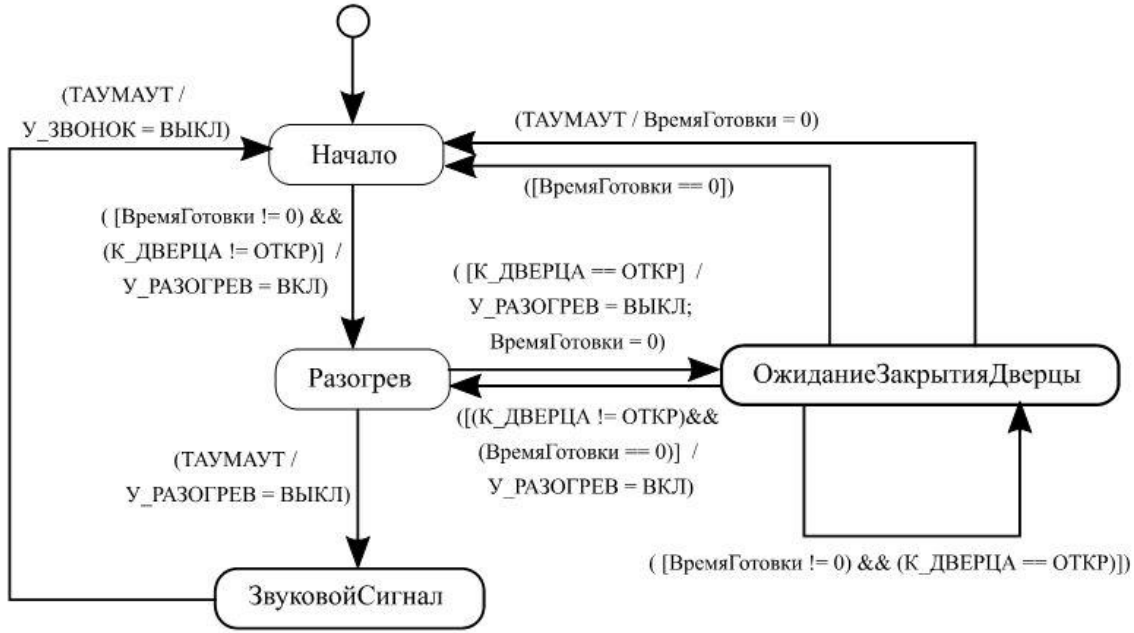


Рис. 6. Диаграмма состояний для процесса Разогрев.

Для отображения состояний процесса подошла диаграмма состояний UML.

Диаграмма связи процессов по данным

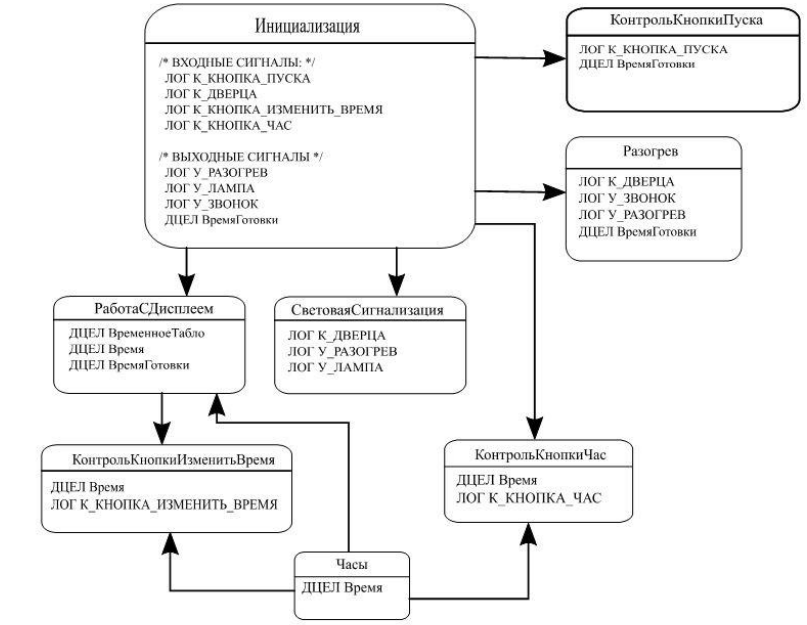


Рис. 4. Диаграмма связи процессов по данным.

Для построения диаграмм связи процессов по данным была проанализирована диаграмма классов UML. Заимствовано:

- Идея отражения зависимости с помощью стрелок.
- Общая структура вершин диаграммы:
  - название процесса, отделенное чертой от тела.
  - описание переменных (в оригинале - полей класса).

Диаграмма связи процессов по управлению

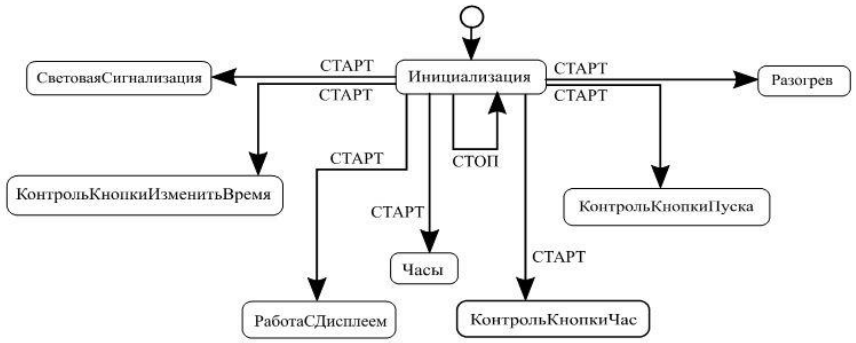


Рис. 5. Диаграмма связи процессов по управлению.

Для построения диаграмм связи процессов по управлению были проанализированы:

- диаграмма деятельности UML:
  - общий вид вершин диаграммы (вместо названий активностей имена процессов;
- диаграмма состояний UML:
  - идея подписей над стрелками;
  - обозначение точки входа.

Прогресс работы: что сделано в прошлом семестре

- Проведен анализ:
  - специфики ПОП на языке Reflex;
  - существующих видов диаграмм, использующихся для анализа кода;
  - средств визуализации диаграмм для языков общего назначения;
- сформулированы требования к создаваемому программному модулю;
- разработаны диаграммы для отображения связей процессов по данным и управлению;
- Написано 32 страницы диплома.

Прогресс работы: что сделано в этом семестре

- Проведен сравнительный анализ средств визуализации графов;
- Спроектирована система:
  - определен формат представления диаграмм;
  - разработана архитектура модуля;
- Написаны тезисы на МНСК.

Обзор средств визуализации графов

Были рассмотрены следующие критерии при анализе:

- ➔ Платно для коммерческого использования
- ➔ Возможность перетаскивания
- ➔ Auto layout
- ➔ Количество алгоритмов укладки графа
- ➔ Формат представления графов
- ➔ Количество поддерживаемых форматов
- ➔ Настройка параметров визуализации
- ➔ Наличие API
- ➔ Узлы как на требуемых диаграммах
- ➔ WYSIWYG***
- ➔ Подписи над ребрами

Сравнительная таблица

GraphViz	-	+*	+	11	dot (gv)	55	+	+	+	-	-
Gephi	-	+	+	12	GEXF, GDF, GML, GraphML, Pajek NET, gv,	9	+	+	-	-	+
					CSV, UCINET DL, Tulip TPL, Netdraw VNA, Spreadsheet						
Igraph	-	-	+	13	Pickle, pajek, net, ncol, GraphMLz, GraphML, gml, edgelist, edges, edge, dl(только чтение), dimacs, adjacency, lgl. Leda, graphviz, dot (только запись)	6	+	+	+	-	+
Graphistry	+	+	+	1	?	?	+	+	-	+	-
OGDF	-	-	+	12	Graph, gml, rome, leda, chaco, PMDissGraph, YGraph, Graph6, MatrixMarket, Rudy, BENCH, PLA, GD-Challenge, GraphML, DOT, GEXF, GDF, TLP, DL, STP, DMF, edgeList	3	+	+	+	-	+

ZGRViewer	-	+	+	11	Dot (gv), svg	?	+	+	+	-	-
Yed	-**	+	+	24+	GraphML, ygf, gml, xgml, xls, xlsx, tgf, ged, XML + XSL, edge list, node list, graphmlz	12	+	-	+	+	+
NetworkX	-	+*	+	11	GML, GraphML, edge list, GIS Shapefile, Pajek, Sparse6, Graph6, YAML, LEDA, JSON, Pickle, GEXF, Adjacency List, gv	8	+	+	+	-	+
Tulip	-	?	+	7	Tlp, gml, csv, gexf	?	+	+	+	?	+
LEDA	+	?	+	?	Gw, GML, LEDA	?	?	+	+	?	+
NetDraw	-	+	+	3+	Nodelist, edgelist, fullmatrix, vna, Pajek, vna, uciNet	3	+	?	+	+	-

* Примечание. Только в dotty

** Примечание. yEd Software License Agreement

*** Примечание. What you see is what you get.

Обзор форматов представления графов

Были рассмотрены следующие критерии при анализе:

- Хранение координат
- Форма вершин
- Ориентированный граф
- Подписи над ребрами
- Поддерживающие средства

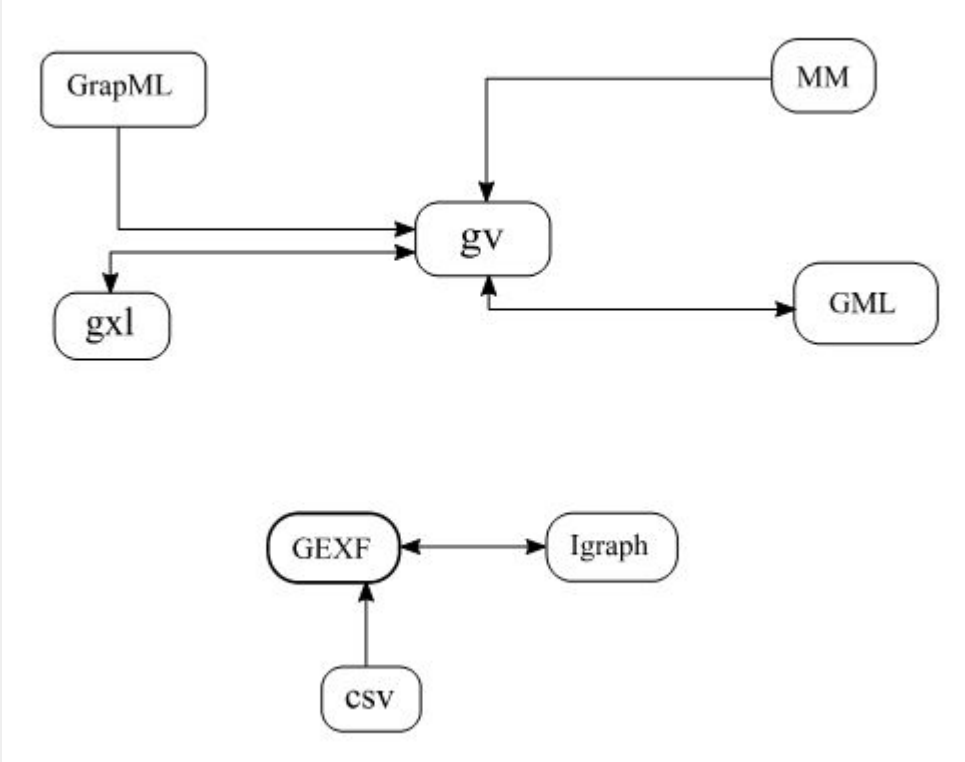
Название	Хранение координат	Форма вершин	Ориентированный граф	Подписи над ребрами	Поддерживающие средства
GraphML [23][24][25]	+	+	+	+	Gephi, Igraph, OGDF, Yed, NetworkX
Gv [26] [27]	-	+	+	+	GraphViz, Gephi, Igraph*, OGDF, ZGRViewer, NetworkX
Xgml* [28]	+	+	+	+	Yed
Gml	+	+	+	+	Gephi, Igraph, OGDF, Yed, NetworkX, Tulip, LEDA
Node list	-	-	+	-	Yed, NetDraw
Edge list	-	-	+	-	OGDF, Yed, NetworkX, NetDraw
Pajek[30]	+	+	+	+	Gephi, Igraph, NetworkX, NetDraw
Leda[31]	+	+	+	+	Igraph, OGDF, NetworkX, LEDA
TLP [32]	+	+	+	+	OGDF, Gephi, Tulip
Gw	+	+	+	+	LEDA
GEXF [40][41]	+	+	+	-	Gephi, OGDF, NetworkX, Tulip

Схема возможных преобразований форматов

Было замечено, что форматы разбиваются на две группы, внутри которых возможна конвертация.

Наибольший интерес представляет верхняя, так в ней находятся наиболее популярные и удовлетворяющие критериям отбора форматы.

Вывод: наиболее подходящими форматами оказались GML и GraphML



Сравнение GML и GraphML

GML и GraphML:

- наиболее поддерживаемые форматы;
- возможности конвертации во множество других форматов;
- удовлетворяют критериям отбора.

Критерий сравнения	GML	GraphML
Диаграмма связи по данным	+	+
Диаграмма связи по управлению	+	+
Диаграмма состояний процесса	+	+
Размер файла	x	2x
Легок для восприятия человеком	+	-
Поддержка русского языка в надписях	+*	+

* Примечание. Необходимо записывать ASCII-коды символов в 16-ричном виде.

Преимущества GML

- Простота восприятия человеком
- Небольшой размер файла
- Широкая поддержка сторонними приложениями
- Возможности конвертации в широкий спектр форматов
- Поддержка русского языка
- Возможность хранения требуемых диаграмм

Реализация: принципы

- На вход модуль получает AST дерево
- путем прохода по AST и его анализу, (возможно, неоднократному), создается:
  - список процессов
  - список переменных для каждого процесса
  - список состояний с пометкой об условиях перехода и прочей информацией, необходимой для диаграмм (откуда вызываются другие процессы и тд.)
- На основании построенной модели, создаются файлы GML диаграмм (output)

Архитектура модуля

Были рассмотрены следующие варианты архитектуры:

- встраивание в имеющийся транслятор языка Reflex
- разработка отдельного приложения на Java
- разработка плагина Eclipse для дальнейшей интеграции с создающимся Reflex IDE.

Встраивание в транслятор

Был разработан прототип, имеющий следующие особенности:

- язык: C++
- AST дерево, созданное транслятором, берется из памяти
- зависимость от транслятора, который скоро станет устаревшим
- невозможность встраивания в Reflex IDE
- написано ~200 строк кода
- input: AST из памяти
- output: GML файлы диаграмм (пример на рисунке)

```
Version "2.15"
graph
[
    hierarchic  1
    label   ""
    directed    1
    node
    [
        id  0
        label   "Init"
        graphics
        [
            w   20.0
            h   48.0
            type    "roundrectangle"
            raisedBorder    0
            fill    "#FFFFFF"
            outline "#000000"
        ]
        LabelGraphics
        [
            text    "Init"
            fontSize    12
            fontName    "Dialog"
            anchor  "c"
        ]
    ]
    node
    [
        id  1
        label   "Init2"
        graphics
        [
            w   25.0
            h   48.0
            type    "roundrectangle"
            raisedBorder    0
            fill    "#FFFFFF"
            outline "#000000"
        ]
        LabelGraphics
        [
            text    "Init2"
            fontSize    12
            fontName    "Dialog"
            anchor  "c"
        ]
    ]
]
```

Разработка отдельного приложения

Был реализован прототип со следующими свойствами:

+ язык: Java
- AST дерево создается транслятором, который сейчас находится в стадии разработки. Взаимодействие предполагалось организовать через сериализацию
- Сложность интеграции с Reflex IDE, реализующейся на базе Eclipse
- Сложность отладки ввиду незавершенности транслятора
- Необходимость согласовать формат дерева

Разработка отдельного приложения

- написано ~ 300 строк кода
- input: AST через сериализацию или отдельный файл
- output: GML файлы диаграмм
- на данном этапе решено отложить работу над этим решением, и попробовать более перспективное с точки зрения дальнейшего развития (Eclipse - плагин)

Разработка плагина Eclipse

+ язык: Java, Xtext
- AST дерево создается транслятором и хранится в виде модели
+ Простота интеграции с Reflex IDE, реализующейся на базе Eclipse
- Сложность отладки ввиду незавершенности транслятора
- Необходимость согласовать формат дерева
- Возможно, средства Xtext не позволяют во время исполнения проходить по AST дереву.