МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ Государственное автономНОЕ образовательное

учреждение высшего образования

«Новосибирский НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ государственный университет»

(нОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

Факультет информационных технологий

Кафедра…компьютерных технологий………………………………………………………….

1. Направление подготовки: 09.03.01 Информатика и вычислительная техника
2. Образовательная программа: 09.03.01 Информатика и вычислительная техника

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА БАКАЛАВРА**

|  |
| --- |
| разработка программного модуля визуализации диаграмм процессов по спецификации на языке Reflex \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (название темы прописными буквами) |
| утверждена распоряжением проректора по учебной работе №0290 от «30» сентября 2019г. |
| \_\_\_ Беленькой Софии Евгеньевны, группа16205\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

(Фамилия, Имя, Отчество студента, группа) (подпись студента)

|  |  |
| --- | --- |
| **«К защите допущена»** | **Руководитель ВКР** |
| Заведующий кафедрой, | д. т. н., доцент, зав. лаб. ИАиЭ СО РАН, |
| д. т. н., доцент, зав. лаб. ИАиЭ СО РАН | Зав. кафедрой КТ |
| Зюбин В.Е /………….. | Зюбин В.Е /………... |
| (ФИО) / (подпись) | (ФИО) / (подпись) |
| «……»………………2020г. | «……»………………2020г. |

Дата защиты: «……»………………2020г.

Новосибирск, 2020г.

# Содержание

[ВВЕДЕНИЕ 5](#_Toc42638066)

[ГЛАВА 1 7](#_Toc42638067)

[1.1 Обзор средств автоматической визуализации диаграмм по коду для языков общего назначения 7](#_Toc42638068)

[1.1.1 Рейтинг IDE. 7](#_Toc42638069)

[1.1.2 Средства построения графической документации для языков общего назначения 7](#_Toc42638070)

[1.1.3 Выводы 11](#_Toc42638071)

[1.2 Обзор средств разработки систем управления и ПО для встраиваемых систем 12](#_Toc42638072)

[1.2.1 Анализ средств разработки систем управления и ПО для встраиваемых систем. 12](#_Toc42638073)

[1.2.2 Вывод. 16](#_Toc42638074)

[1.3 Анализ специфики ПОП на примере языка Reflex 16](#_Toc42638075)

[1.3.1 Процесс-ориентированный язык Reflex. 16](#_Toc42638076)

[1.3.2 Выводы. 18](#_Toc42638077)

[ГЛАВА 2 19](#_Toc42638078)

[2.1 Требования к программному модулю 19](#_Toc42638079)

[2.2 Разработка диаграмм для визуализации связей процессов 19](#_Toc42638080)

[2.2.1 Диаграмма состояний процесса. 19](#_Toc42638081)

[2.2.2 Диаграмма связи процессов по данным. 20](#_Toc42638082)

[2.2.3 Диаграмма связи процессов по управлению. 22](#_Toc42638083)

[2.3 Выбор формата представления диаграмм 22](#_Toc42638084)

[2.3.1 Обзор средств визуализации графов. 22](#_Toc42638085)

[2.3.2 Обзор форматов хранения графов. 27](#_Toc42638086)

[2.3.3 Сравнение GML и GraphML. 28](#_Toc42638087)

[2.3.4 Выводы. 29](#_Toc42638088)

[ГЛАВА 3 30](#_Toc42638089)

[3.1 Архитектура модуля 30](#_Toc42638090)

[3.2 Реализация модуля 30](#_Toc42638091)

[3.2.1 Варианты реализации модуля 30](#_Toc42638092)

[3.2.2 Реализация модуля разделения моделей диаграмм без общих вершин 31](#_Toc42638093)

[3.3 Алгоритм использования модуля 33](#_Toc42638094)

[ЗАКЛЮЧЕНИЕ 34](#_Toc42638095)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ 36](#_Toc42638096)

[Список публикаций автора по теме выпускной квалификационной работы 44](#_Toc42638097)

[ПРИЛОЖЕНИЕ А программа управления микроволновой печью 45](#_Toc42638103)

[ПРИЛОЖЕНИЕ Б тезисы МНСК 51](#_Toc42638104)

[ПРИЛОЖЕНИЕ В Программный модуль генерации диаграмм процессов по спецификации на языке Reflex Руководство оператора 52](#_Toc42638105)

[АННОТАЦИЯ 55](#_Toc42638106)

[1 Назначение программного модуля 56](#_Toc42638107)

[1.1. Функциональное назначение программного модуля 56](#_Toc42638108)

[1.2. Эксплуатационное назначение программного модуля 56](#_Toc42638109)

[1.3. Функции модуля 56](#_Toc42638110)

[2 Условия выполнения программы 56](#_Toc42638111)

[2.1. Минимальный состав аппаратных средств 56](#_Toc42638112)

[2.2. Минимальный состав программных средств 57](#_Toc42638113)

[2.3. Требования к оператору 57](#_Toc42638114)

[3 Выполнение программы 57](#_Toc42638115)

[3.1. Загрузка и запуск программного модуля 57](#_Toc42638116)

[3.2. Выполнение программного модуля 57](#_Toc42638117)

[3.3. Завершение работы программного модуля 57](#_Toc42638118)

[4 Сообщения оператору 57](#_Toc42638119)

[5 Лист регистрации изменений 58](#_Toc42638120)

[ПРИЛОЖЕНИЕ Г Программный модуль визуализации диаграмм процессов по спецификации на языке Reflex Описание программы 59](#_Toc42638121)

[Cодержание 60](#_Toc42638122)

[АННОТАЦИЯ 61](#_Toc42638123)

[2 Общие сведения 62](#_Toc42638124)

[1.1 Обозначение и наименование программного модуля 62](#_Toc42638125)

[1.2 Программное обеспечение, необходимое для функционирования программного модуля 62](#_Toc42638126)

[1.3 Языки программирования 62](#_Toc42638127)

[3 Функциональное назначение 63](#_Toc42638128)

[2.1 Назначение программного модуля 63](#_Toc42638129)

[4 Описание логической структуры 64](#_Toc42638130)

[3.1 Структура программного модуля 64](#_Toc42638131)

[3.2 Алгоритм программного модуля 69](#_Toc42638132)

[3.3 Связи между составными частями программы 69](#_Toc42638133)

[3.4 Связи программы с другими программами 69](#_Toc42638134)

[4 Используемые технические средства 69](#_Toc42638135)

[5 Вызов и загрузка 70](#_Toc42638136)

[6 Входные данные 70](#_Toc42638137)

[7 Выходные данные 70](#_Toc42638138)

[8 Лист регистрации изменений 70](#_Toc42638139)

# ВВЕДЕНИЕ

Существует несколько моделей разработки программного обеспечения, и в последнее время наибольшей популярностью пользуется итерационная модель разработки, при работе с которой возникает необходимость создания текущей рабочей документации, в частности, построении различных диаграмм по коду. Также подобная необходимость возникает при реверсивном инжиниринге и создании новых версий существующего программного продукта, так как диаграммы позволяют наглядно описать программу в выходной документации.

Для популярных языков общего назначения (таких, как Java, C++, С# и другие) реализовано множество программных средств автоматического построения диаграмм по коду, часть из которых встроены в интегрированные среды разработки (IDE). В то же время, для промышленной автоматизации, при написании управляющих алгоритмов и программировании встраиваемых систем вопрос разработки средств анализа исходного кода стоит очень остро: здесь используются либо проблемно-ориентированные языки (DSL), либо языки общего назначения в роли низкоуровневого средства. В том числе эта проблема актуальна и для активно развивающегося процесс-ориентированного программирования (ПОП) [1][2][3], которое разрабатывается в Институте автоматики и электрометрии. В текущих проектах анализ исходного кода делается вручную, что очень трудоемко, а создаваемая при этом документация содержит ошибки. В связи с этим для ПОП остро стоит задача автоматизации процесса создания графической документации, а именно, построения диаграмм по исходному коду программы. В качестве языка программирования в работе рассматривался язык Reflex [2][4][5][6][7].

Цель работы: разработка программного модуля визуализации диаграмм процессов по спецификациям на языке Reflex. В рамках означенной цели поставлены следующие задачи:

- провести анализ:

* 1. специфики ПОП на языке Reflex с точки зрения необходимости визуализации диаграмм;
  2. средств визуализации диаграмм по коду для языков общего назначения и для специализированных языков;

- спроектировать систему:

1. сформулировать требования к создаваемому программному модулю;
2. разработать диаграммы для отображения связей процессов;
3. определить формат представления диаграмм;

- разработать архитектуру модуля, удовлетворяющую обозначенным требованиям;

- реализовать модуль визуализации;

- провести тестирование созданной реализации, опробовать ее на практике.

При построении диаграмм будет использоваться синтаксическое дерево разбора, построенное IDE или сторонним приложением. Таким образом, обеспечивается возможность встраивания модуля в состав IDE помимо возможности его независимой работы.

Автоматическая визуализация диаграмм позволит значительно сократить время создания графической документации, что упростит разработку и поддержку проектов на языке Reflex.

Работа состоит из трех глав, в первой из которых приведен анализ специфики ПОП на примере языка Reflex и обзор средств автоматической генерации графической документации по исходному коду программ. Во второй главе сформулированы требования к создаваемому программному модулю, разработаны нотации диаграмм для отображения связей процессов и определен формат представления диаграмм. В третьей главе описывается разработанная архитектура модуля и его реализация, а также результаты тестирования разработанного программного модуля на практике.

# ГЛАВА 1

## Обзор средств автоматической визуализации диаграмм по коду для языков общего назначения

### Рейтинг IDE.

Рассмотрим рейтинг IDE, составленный на основе информации о частоте запросов загрузочной страницы в поисковике Google в 2019 году [8] (см. Таблицу 1).

Таблица 1 - top IDE index

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Rank** | **Change** | **IDE** | **Share** | **Trend** |
| 1 | http://pypl.github.io/Up.pnghttp://pypl.github.io/Up.png | Android Studio | 23.38 % | +8.8 % |
| 2 | http://pypl.github.io/Down.png | Visual Studio | 21.81 % | -2.1 % |
| 3 | http://pypl.github.io/Down.png | Eclipse | 18.1 % | -5.3 % |
| 4 |  | NetBeans | 5.54 % | -0.5 % |
| 5 | http://pypl.github.io/Up.pnghttp://pypl.github.io/Up.pnghttp://pypl.github.io/Up.png | pyCharm | 4.89 % | +1.0 % |
| 6 | http://pypl.github.io/Up.pnghttp://pypl.github.io/Up.pnghttp://pypl.github.io/Up.png | Visual Studio Code | 4.88 % | +1.0 % |
| 7 | http://pypl.github.io/Down.pnghttp://pypl.github.io/Down.png | IntelliJ | 4.8 % | -0.0 % |
| 8 | http://pypl.github.io/Up.pnghttp://pypl.github.io/Up.png | Xcode | 3.91 % | +0.5 % |
| 9 | http://pypl.github.io/Down.pnghttp://pypl.github.io/Down.pnghttp://pypl.github.io/Down.png | Sublime Text | 3.66 % | -0.7 % |
| 10 | http://pypl.github.io/Down.pnghttp://pypl.github.io/Down.pnghttp://pypl.github.io/Down.png | Atom | 3.2 % | -0.9 % |
| 11 |  | Code::Blocks | 1.38 % | -0.4 % |
| 12 | http://pypl.github.io/Up.png | Vim | 0.89 % | -0.2 % |
| 13 | http://pypl.github.io/Down.png | Xamarin | 0.75 % | -0.4 % |

Проанализируем данный рейтинг с точки зрения наличия возможности построения UML-диаграмм. Зеленым цветом помечены строки с теми средствами разработки, в которых такая возможность есть, и красным – те, в которых она отсутствует. Из таблицы видно, что первая десятка популярных IDE в большинстве своем имеет встроенную возможность построения UML-диаграмм. Отсюда можно сделать вывод, что данный инструмент является востребованным. Рассмотрим подробно различные средства их построения.

### Средства построения графической документации для языков общего назначения

Рассмотрим средства для построения графической документации по коду для популярных языков общего назначения.

1. Утилита Class Designer для Visual Studio [9] способна динамически генерировать диаграмму классов по исходному коду, производить реверсивный инжиниринг и отслеживать наследственные связи с другими классами, а также реализует возможность перетаскивания элементов диаграммы и настройки видимости полей классов на ней. Для коммерческого использования юридическим лицом необходима платная лицензия при условии создания проекта с закрытым кодом. Visual Studio поддерживает широкий спектр языков и технологий, которые могут быть встроенными или предоставляться в виде расширений, в частности: C#, F#, Visual Basic, C++, Python, JavaScript, PHP, HTML, CSS, JavaScript и JSON.
2. Бесплатная программная платформа Astah UML[10] позволяет создавать следующие диаграммы: Class diagram (Object, Package, Subsystem and Robustness Diagrams are included), Use case diagram, Sequence diagram, Collaboration Diagram, State diagram, Activity diagram, Deployment diagram, Component diagram, способна генерировать Java 1.4 sourcecode из модели, а также импортировать Java 1.4 source files для генерации модели.
3. MagicDraw [11] - визуальный UML , SysML , BPMN и UPDM инструмент моделирования, предоставляет возможности реверсивного инжиниринга, кодогенерации для следующих языков программирования: J2EE, C#, C++, CORBA IDL programming languages, .NET, XML Schema, WSDL, Java, EJB, DDL. Создает class, deployment, object, profile, package diagram, composite structural diagram, и другие. Требует платной лицензии.
4. Software Ideas Modeler [12] - легкий и мощный инструмент для создания диаграмм UML и некоторых других диаграмм. Он поддерживает все 14 типов диаграмм UML, BPMN 2.0, SysML, ArchiMate, JSD, HTA, смешанную диаграмму, диаграмму потока данных, диаграмму последовательности операций, диаграмму надежности, диаграмму отношений сущностей, диаграмму требований, моделирование пользовательского интерфейса, диаграмму карты CRC, диаграмму параллелизма, дерево поведения, структурную диаграмму и т.д. Генерирует код для ActionScript, C#, C++, Delphi (Object Pascal), Java, JavaScript, PHP, Python, Ruby, SQL, Visual Basic, Visual Basic .NET, XML Schema, и осуществляет реверсивный инжиниринг для C#, C++, Java, PHP, Ruby, Visual Basic .NET. Требует платной лицензии для коммерческого использования.
5. BOUML [13] – бесплатный конструктор UML-диаграмм, поддерживающий реверсивный инжиниринг для языков C++, Java, PHP, MYSQL, и кодогенерацию из диаграммы классов в них, а также в Python и IDL. Программа может строить диаграммы классов, последовательности, кооперации, объектов, прецедентов, компонентов, состояний, деятельности, диаграммы компонент и схем развертывания.
6. Пакет инструментов моделирования бизнес-процессов, ПО и баз данных Visual paradigm [14] поддерживает 14 видов диаграмм: Class diagram, Use case diagram, Sequence diagram, Communication diagram, State machine diagram, Activity diagram, Component diagram, Deployment diagram, Package diagram, Object diagram, Composite structure diagram, Profile diagram, Timing diagram, Interaction overview diagram. Требует платной лицензии. Способен генерировать код для Java, C++, CORBA IDL, PHP, XML Schema, Ada, Python, C#, VB .NET, Object Definition Language (ODL), Flash ActionScript, Delphi, Perl, Objective-C и Ruby. Также осуществляет реверсивный инжиниринг для Java, C++, CORBA IDL, PHP, XML Schema, Ada, Python, C#, Java class, .NET dll и exe, JDBC.
7. Rational Rose [15] представляет собой набор инструментов моделирования UML для разработки программного обеспечения. Есть функции кодогенерации и реверсивного инжиниринга для таких языков, как C++, Ada, Java/J2EE, Visual C++, Visual Basic и XML. Также существует возможность построения следующих диаграмм: Use Case diagram, Logical diagram, Component diagram, Deployment diagram, State diagram, Activity diagram, Iteration diagram. Больше не поддерживается.
8. Enterprise Architect [16] – инструмент дизайна и визуального моделирования. Поддерживается кодогенерация и реверсивный инжиниринг для ActionScript, С, C #,C ++, Delphi, Java, PHP, Python, Visual Basic, Visual Basic .NET. ПО работает со следующими диаграммами: диаграмма классов, объектов, составная структурная диаграмма, диаграмма компонент, размещения, пакетов, взаимодействия, деятельности, функций, состояний, последовательностей, обзорные диаграммы потоков управления, коммуникационные диаграммы, временные диаграммы. Требует платной лицензии.
9. IntelliJ Idea [17] – платная IDE, поддерживающая такие языки, как Java, Scala, Groovy, Kotlin, JavaScript, TypeScript, SQL и способная создавать диаграмму классов по коду. Также среда поддерживает module dependencies diagram.
10. Sybase PowerDesigner [18] - средство UML проектирования с поддержкой кодогенерации для Java, C++, C#, VB .NET, Hibernate, EJB3, NHibernate, JSF, WinForm (.NET and .NET CF), PowerBuilder, также присутствуют возможности реверсивного инжиниринга для этих языков, исключая C++. Строит use-case, activity, sequence, диаграммы классов и компонентов, а также многое другое. Требует платной лицензии.
11. NetBeans [19] - свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, Python, PHP, JavaScript, C, C++, Ада, и ряда других. Поддерживаются следующие виды диаграмм: activity, class, sequence, state, use-case diagram, есть возможности реверсивного инжиниринга.
12. LabVIEW [20] - это среда разработки и платформа для выполнения программ. Позволяет генерировать диаграммы классов. Поддерживает кодогенерацию в С-код. Требует платной лицензии.
13. NClass [21] – это проект с открытым кодом. Поддерживает диаграммы классов и кодогенерацию для языков C# и Java. Полностью бесплатен.
14. Altova UModel 2008 [22] – инструмент моделирования UML-диаграмм. Поддерживает кодогенерацию и реверсивный инжиниринг, генерацию документации для Java, C++, C#, Visual Basic и .NET. Требует платной лицензии. Работает с Use case diagrams, Class, object, Composite structure diagrams, Component, Deployment, Package, Profile diagram, Sequence, Activity, State machine diagrams, Interaction overview diagrams, Timing, Communication diagrams.

Более наглядно сравнение средств построения UML-диаграмм приведено в таблице 2. Зеленым цветом выделены критерии, которые рассматриваемое средство поддерживает, красным – которые не поддерживает. Розовым цветом выделены критерии, информация по поддержке которых для данного средства не была найдена.

Таблица 2 – сравнение средств построения UML-диаграмм

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Реверсивный инжиниринг** | **Кодогенерация** | **Возможность модификации** | **Динамическое построение** | **Возможность скрывать компоненты классов** | **Количество диаграмм** |
| Class Designer для Visual Studio | + | - | + | + | + | 1? |
| Astah UML | + | + | + |  |  | 9 |
| MagicDraw | + | + | + | + \* |  | 24 |
|
| Software Ideas Modeler | + | + | + | +? | + | 14 |
| BOUML | + | + | + | -? | + | 10+ |
| Visual paradigm | + | + | + |  |  | 28 |
| Rational Rose | + | + | + | + | +? | 5+ |
| Enterprise Architect. | + | + | + |  |  | 12 |
| IntelliJ Idea | + | - | + | +? | + | 3 |
| Sybase PowerDesigner | + | + | + | -? | + | 3+ |
| NetBeans | + | - | + | -? | -? | 5+ |
| Lab view | + | + | + |  | - | 1 |
| Altova UModel 2008 | + | + | + |  |  | 14+ |

### Выводы

Были рассмотрены наиболее известные средства построения UML-диаграмм для языков общего назначения, таких, как Java, C# и другие. Исходя из специфики поставленной задачи, наибольший интерес представляют продукты, способные осуществлять реверсивный инжиниринг, так как в разрабатываемом программном модуле предполагается генерация диаграмм по коду. Средства построения, не удовлетворяющие данному критерию, не рассматривались.

Из таблицы 2 видно, что наиболее широкий спектр возможностей предоставляют такие программные продукты, как Class Designer для Visual Studio, Rational Rose и IntelliJ Idea, двое из которых также находятся в первой семерке популярности по количеству поисковых запросов в Google их загрузочной страницы. Отсюда можно сделать вывод, что наличие в IDE такого средства анализа кода, как визуализация диаграмм, дает преимущества и упрощает разработку. Все рассмотренные средства реализуют возможности модификации построенных диаграмм (drag-and-drop, изменение подписей и другие). Возможности скрывать компоненты отдельных классов и динамическое построение реализуется лишь частью рассмотренных средств.

## Обзор средств разработки систем управления и ПО для встраиваемых систем

### Анализ средств разработки систем управления и ПО для встраиваемых систем.

Рассмотрим наиболее популярные средства разработки систем управления и ПО для встраиваемых систем с точки зрения поддержки реверсивного инжиниринга и генерации графической документации. Более наглядно результаты сравнения представлены в таблице 3. Зеленым цветом выделены критерии, которые рассматриваемое средство поддерживает, красным – которые не поддерживает. Розовым цветом выделены критерии, информация по поддержке которых для данного средства не была найдена.

1. MATLAB: это высокоуровневый язык и интерактивная среда для программирования, численных расчетов и визуализации результатов. Требует платной лицензии. MATLAB Coder работает с Simulink Coder и Embedded Coder для генерации C кода из моделей Simulink, которые содержат код MATLAB.
2. Stateflow - это среда для моделирования и симуляции комбинаторной и последовательной логики принятия решений, основанных на машинах состояний и блок-схемах. Матрица перехода состояний генерируется из таблицы перехода состояний. Используя расширения для генерации кода, вы можете генерировать код PLC, C и C++ или HDL непосредственно из вашей диаграммы состояний.
3. CoDeSys - инструментальный программный комплекс промышленной автоматизации. Распространяется бесплатно демонстрационная версия, полная платно. Поддерживает IL, ST, LD, FBD, SFC, CFC. Поддерживаются основные 16- и 32-разрядные процессоры: Infineon C166, TriCore, 80x86, ARM (архитектура), PowerPC, SH, MIPS (архитектура), Analog Devices Blackfin, TI C2000/28x и другие. В качестве UML-редакторов интегрируются два типа диаграмм – классов (Class Diagram) и состояния оборудования (State Machine Diagram).
4. ISaGRAF — инструментальная графическая среда разработки прикладных программ для программируемых логических контроллеров на языках стандарта IEC 61131-3 и IEC 61499, позволяющая создавать локальные или распределенные системы управления. Поддерживает кодогенерацию для ISA86M, ISA68M, SCC, CC86M (т.е. С-код).
5. YAPLC - свободная система программирования ПЛК. Для программирования используются пять языков стандарта IEC-61131-3, для расширения программ можно использовать С. В состав входят Beremiz - интегрированная среда разработки программных ПЛК на языках IEC-61131-3; matiec - транслятор языков програмрования IEC-61131-3, генерирует программный ПЛК на С; GNU ARM Embedded Toolchain - набор инструментов разработчика на С и С++.
6. LogicLab [24] – программа, позволяющая симулировать и отображать логические условия, используя логические операции, сегментационные декодеры, бинарное кодирование и декодирование, ALU и магнитудное сравнение. Программа предназначается для проведения логических операций на основе событий, которые происходят. При помощи LogicLab пользователь может задать цепочку действий, которые нужно предпринять в зависимости от наступления тех или иных событий. При этом логические данные могут быть визуализированы в форме сегментационного отображения. Генерация машинного кода высокой эффективности для наиболее распространенных процессоров на рынке.
7. SOFTLOGIC в TRACE MODE® - это мощная система программирования промышленных контроллеров, полностью интегрированная со SCADA/HMI. Поддержка языков программирования стандарта IEC МЭК 6-1131/3. Полная версия платная.
8. SIMATIC STEP 7 — программное обеспечение фирмы Siemens для разработки систем автоматизации на основе программируемых логических контроллеров Simatic S7-300/S7-400/M7/C7 и WinAC. Поддерживаются LAD, FBD, STL, SCL, GRAPH 7, HiGraph 7, SFC. Требует платной лицензии.
9. TRACE MODE - программный комплекс класса SCADA HMI. Предназначен для разработки программного обеспечения АСУТП, систем телемеханики, автоматизации зданий, систем учёта электроэнергии (АСКУЭ, АИИС КУЭ), воды, газа, тепла, а также для обеспечения их функционирования в реальном времени. Обладает функциями программирования промышленных контроллеров. Полная версия платная. Встроена генерация документов. Шаблоны документов создаются в удобном визуальном редакторе Интегрированной среды разработки TRACE MODE. Редактор позволяет осуществлять форматирование текста, работать со списками, таблицами, вставлять растровые изображения, тренды, столбчатые и круговые гистограммы и задавать для них тип кривых, цвет фона, линии сетки, масштаб по осям Х и Y, колонтитулы, легенду и т.п.
10. MasterSCADA — программный пакет для проектирования систем диспетчерского управления и сбора данных с возможностью вывода ретроспективы на графики. Построение графиков зависимости параметра от времени, параметра от параметра. Режим просмотра графиков совмещенный с журналом событий, возможность навигации по графикам как по времени, так и по событиям.
11. KB-IDE - это открытая система платы ESP32. поддерживает визуальное программирование, программирование на Arduino. Система с открытым исходным кодом. При блочном кодировании генерируется код на С/С++.
12. Arduino IDE – бесплатная IDE. Программирование осуществляется на приближенном к C/C++ языке программирования Wiring.
13. IDE компании Cygnal (MCS-51), работает с языками С и ассемблер. В окне дизассемблера в пошаговом режиме отображаются команды на языке 8051 ассемблера при исполнении программы, написанной на С.
14. Atmel Studio [23] – основанная на Visual Studio бесплатная проприетарная интегрированная среда разработки (IDE) для разработки приложений для 8- и 32-битных микроконтроллеров семейства AVR и 32-битных микроконтроллеров семейства ARM от компании Atmel. Программа позволяет работать с ассемблером, C, C++. Распространяется бесплатно.
15. CodeVisionAVR [23] – интегрированная среда разработки приложений для AVR микроконтроллеров. Работает с ассемблером и С. Полная версия платная. Содержит генератор начального кода программы, позволяющего произвести инициализацию периферийных устройств.
16. Keil MDK–ARM (Keil uVision) [23] – среда разработки, представляющая собой набор утилит для выполнения полного комплекса мероприятий по написанию приложений для микроконтроллеров. Работает с С или ассемблером. Полная версия платная.
17. Atollic TrueSTUDIO [23] – интегрированная среда разработки программ для ARM микроконтроллеров на базе Eclipse. Поддерживает языки C/C++ и ассемблер. Полная версия платная.
18. IAR Embedded Workbench [23] – среда разработки приложений для целого ряда микроконтроллеров. Позволяет писать программы на языках C, C++ и ассемблере. Полная версия платная.
19. CooCox CoIDE [23] – Высокоинтегрированная программная среда, предназначенная для разработки кода микроконтроллеров архитектуры ARM. Программа основана на базе Eclipse и имеет все ее достоинства. Данная среда разработки совершенно бесплатна и имеет открытый код.
20. Scilab - пакет прикладных математических программ, предоставляющий открытое окружение для инженерных (технических) и научных расчётов. Это самая полная общедоступная альтернатива MATLAB. Содержит интерфейс к Fortran, Tcl/Tk, C, C++, Java, LabVIEW. Распространяется бесплатно.
21. Quantium platform (qm) предназначен для моделирования диаграмм состояний UML, поддерживает кодогенерацию на C или C ++.

Таблица 3 - средства разработки систем управления и ПО для встраиваемых систем

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Название** | **Реверсивный инжиниринг** | **Кодогенерация** | **Возможность модификации диаграмм** | **Платно для коммерческого использования** |
| MATLAB: Stateflow, Simulink | -? | + |  | + |
| CoDeSys | -? | -? |  | + |
| ISaGRAF | -? | + |  | + |
| YAPLC | -? | + |  | -? |
| AXEl Logiclab | +? | + | ? | +? |
| SOFTLOGIC TRACE MODE | - | -? |  | + |
| SIMATIC STEP 7 | - | - |  | + |
| SCADA TRACE MODE | + | - | + | + |
| MasterSCADA | + | -? | + | + |
| KB-IDE | -? | + |  | - |
| Arduino IDE | - | -? |  | - |
| IDE компании Cygnal (MCS-51) | -? | + |  | -? |
| Atmel Studio | -? | -? |  | - |
| CodeVisionAVR | -? | + \* |  | + |
| Keil MDK–ARM (Keil uVision) | -? | -? |  | + |
| Atollic TrueSTUDIO | +? | +? | + | - |
| IAR Embedded Workbench | -? | +? |  | + |
| CooCox CoIDE | +? | +? | + | - |
| Scilab | +? | + | + | - |
| Quantium platform (qm) | -? | + |  | -? |

### Вывод.

Проведенный анализ показывает, что среди средств разработки систем управления и ПО для встраиваемых систем очень малый процент средств разработки предоставляет возможность реверсивного инжиниринга, в том числе это справедливо для средств, предназначенных для коммерческого использования. В основном, реверсивный инжиниринг поддерживают объектно-ориентированные SCADA-пакеты, которые лишь условно относятся к рассматриваемой области.

## Анализ специфики ПОП на примере языка Reflex

### Процесс-ориентированный язык Reflex.

Программа на языке Рефлекс описывается в текстовом виде. Как и в языке Си, символы пробела, табуляции, перевода строки и комментарии – это разделительные символы, которые используются для формирования удобочитаемого листинга программы. Несколько имен, чисел и (или) резервированных слов, расположенных последовательно, должны быть отделены, по крайней мере, одним разделительным символом, чтобы транслятор мог распознавать их как различные объекты. Комментарий в языке Рефлекс начинается последовательностью из символов '/' и '\*' и заканчивается символами '\*' и '/'.

За резервированным словом начала программы “program” указывается имя программы, далее следует открывающая фигурная скобка начала тела программы, которое заканчивается закрывающей фигурной скобкой. Тело программы состоит из двух частей: спецификации общих атрибутов гиперпроцесса (периода цикла, констант, разрешенных для вызова Си-функций, портов ввода/вывода) и описания процессов. В первой части сначала указывается период цикла активизации процессов (в миллисекундах), затем, с помощью резервированного слова const, определяются константы и их значения.

После этого производится спецификация портов ввода / вывода. Порты ввода/вывода рассматриваются как отдельные компоненты языка, которые служат для задания привязки переменных к интерфейсной аппаратуре – устройствам связи с объектом (УСО). Порты различаются по типу (входной или выходной), по размеру и адресу. Указывается тип порта (“input” или “output”), идентификатор порта, два числовых значения, определяющие его системный адрес, и разрядность (8 или 16).

Описание процессов – вторая и основная часть программы, определяющая структуру и поведение управляющего алгоритма, представленного в виде гиперпроцесса. Процессы описываются последовательно. Процесс, описанный первым, является начальным выделенным процессом, т. е. по запуску программы это единственный процесс, находящийся в активном состоянии (в начальном состоянии). Все остальные процессы находятся в пассивном состоянии.

Событийный полиморфизм процесса задается путем последовательного описания его функций-состояний (или далее - просто состояний). Описание состояния начинается с резервированного слова “state”, за которым следуют имя состояния и фигурная скобка, открывающая описание тела состояния, состоящего из событий и реакций на события. События и реакции формируются с помощью условных и безусловных операторов. Описание состояния завершается закрывающей фигурной скобкой.

Стандартные Си-операторы расширены операторами, позволяющими организовать совместное функционирование процессов во времени:

1. операторы запуска/останова процесса – конструкции start <имя процесса>, stop <имя процесса>;
2. операторы управления состоянием текущего процесса – конструкции set state <имя состояния>, set next;
3. оператор контроля времени timeout <спецификатор времени/количество тактов>.

Пример программы управления микроволновой печью представлен в приложении А. Имеющийся в пособии [8] код был дополнен реализацией часов, показывающих текущее время, когда микроволновая печь не используется, а также добавлена возможность задания времени. Кроме того, было изменено поведение алгоритма при открытии дверцы во время разогрева: теперь время готовки не сбрасывается, и при закрытии дверцы разогрев продолжается до конца заданного времени. Если же на протяжении 30 минут дверца не была закрыта, время разогревания сбрасывается, и система переходит в начальное состояние.

### Выводы.

Программа на языке Reflex состоит из описания процессов, начинающихся с ключевого слова proc. Процессы представлены автоматами состояний. Состояния описываются сходными с языком Си конструкциями. При описании процесса можно указать импортируемые из других процессов переменные через ключевые слова from proc. Процессы могут запускать и останавливать другие процессы с помощью ключевых слов start и stop. Исполнение процессов происходит в рамках кооперативной модели многопоточности.

# ГЛАВА 2

## Требования к программному модулю

На основании проведенного анализа, был сформулирован ряд требований к создаваемому программному модулю. Необходимо обеспечить:

1. возможность редактирования диаграмм сторонними средствами;
2. автоматическую укладку диаграмм на плоскость сторонними средствами;
3. визуализацию
4. диаграмм состояний процесса;
5. диаграмм связи процессов по данным;
6. диаграмм связи процессов по управлению;
7. возможность автоматического разделения диаграмм, не имеющих общих вершин, на разные файлы.
8. возможность сохранения диаграммы в отдельный файл.
9. Возможность хранения ссылок на диаграммы состояний процессов внутри вершин процессов остальных диаграмм

## Разработка диаграмм для визуализации связей процессов

### Диаграмма состояний процесса.

Диаграмму состояний процесса предложено отображать в виде диаграммы состояний UML [25][26][27]. Пример такой диаграммы для процесса Cooking программы управления микроволновкой (см. приложение А) представлен на рис. 1. Внутри вершин расположены названия состояний, соединенные стрелками переходов, над которыми указывается в квадратных скобках ограждающее выражение, т. е. условие перехода. После косой черты указываются действия, выполняемые при переходе в данное состояние.

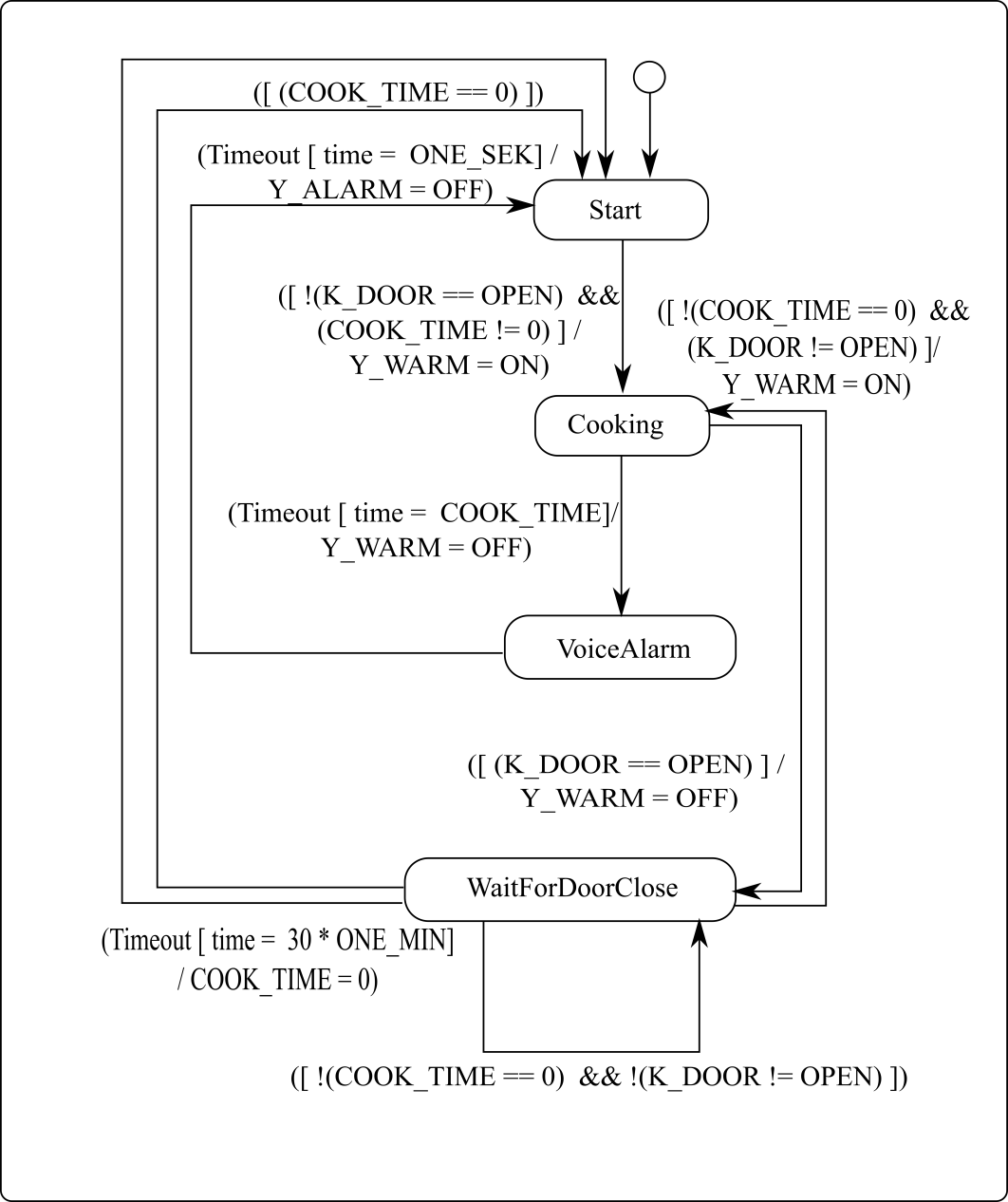


Рисунок 1 - диаграмма состояний для процесса Cooking

### Диаграмма связи процессов по данным.

Для построения диаграмм связи процессов по данным предложено использовать нотацию, разработанную на основе модифицированной диаграммы деятельности и состояний UML [25][26][27]. При этом из диаграммы состояний взята идея обозначений вершин процессов, а из диаграммы деятельности – идея обозначения стрелок связи процессов с переменными и форма вершин переменных.

В вершинах прямоугольной формы отображаются названия процессов, соединенных с овальными вершинами, внутри которых указаны названия разделяемых переменных. Над стрелкой подписывается тип взаимодействия – импортирование (с ключевым словом import) или объявление (т. е. declare).

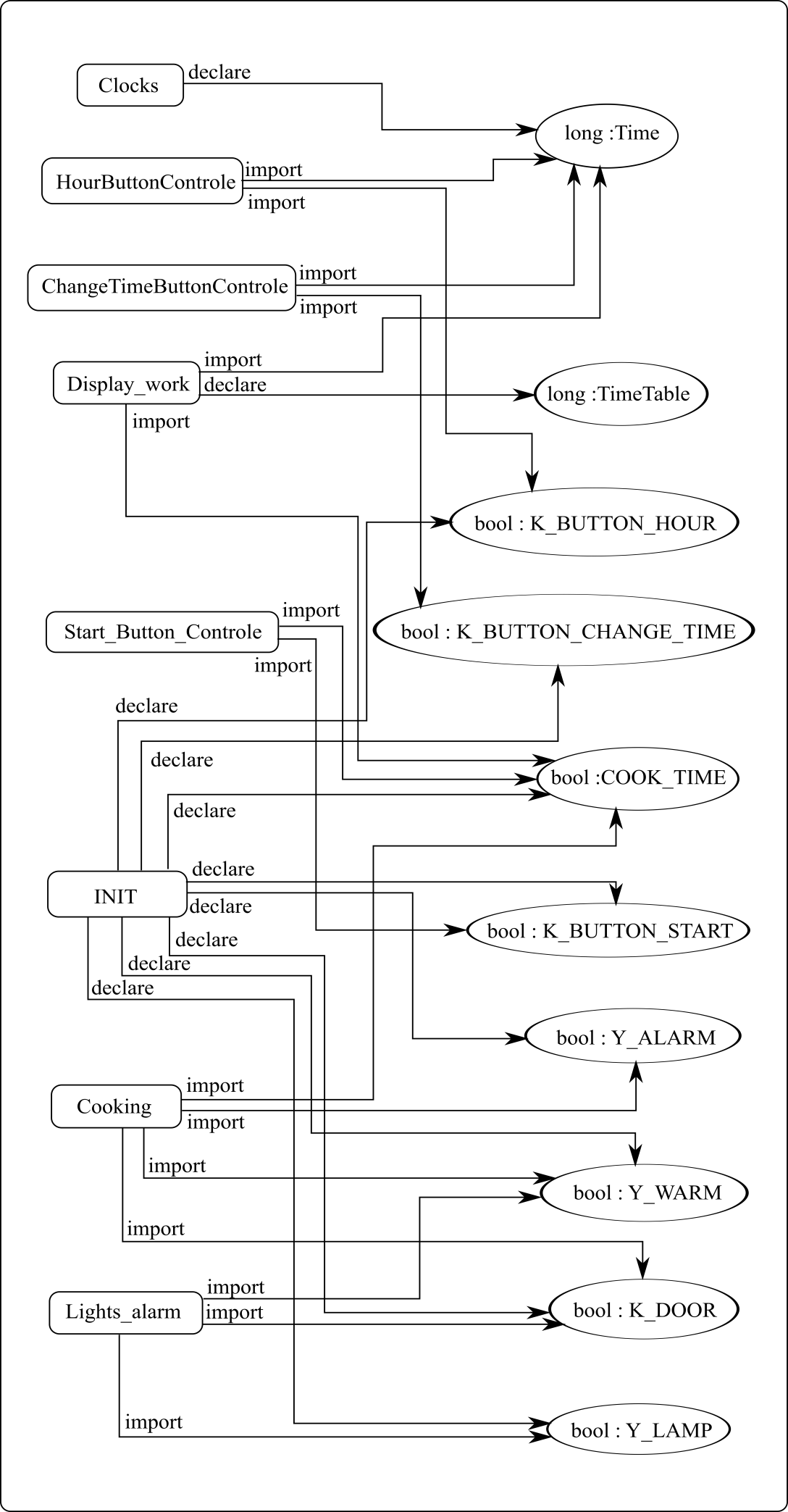


Рисунок 2 - диаграмма связи процессов по данным

### Диаграмма связи процессов по управлению.

Для построения диаграмм связи процессов по управлению были модифицированы:

1. диаграмма деятельности UML;
2. диаграмма состояний UML[25][26][27].

Из диаграммы деятельности был взят общий вид вершин диаграммы. Из нотации диаграммы состояний UML была использована идея подписей над стрелками и обозначение точки входа в программу. Пример диаграммы связи процессов по данным представлен на рис. 3.

Внутри вершин указаны названия процессов. Вершины соединены стрелками, отражающими связь по управлению, причем тип связи указан над стрелкой с помощью ключевых слов start и stop.

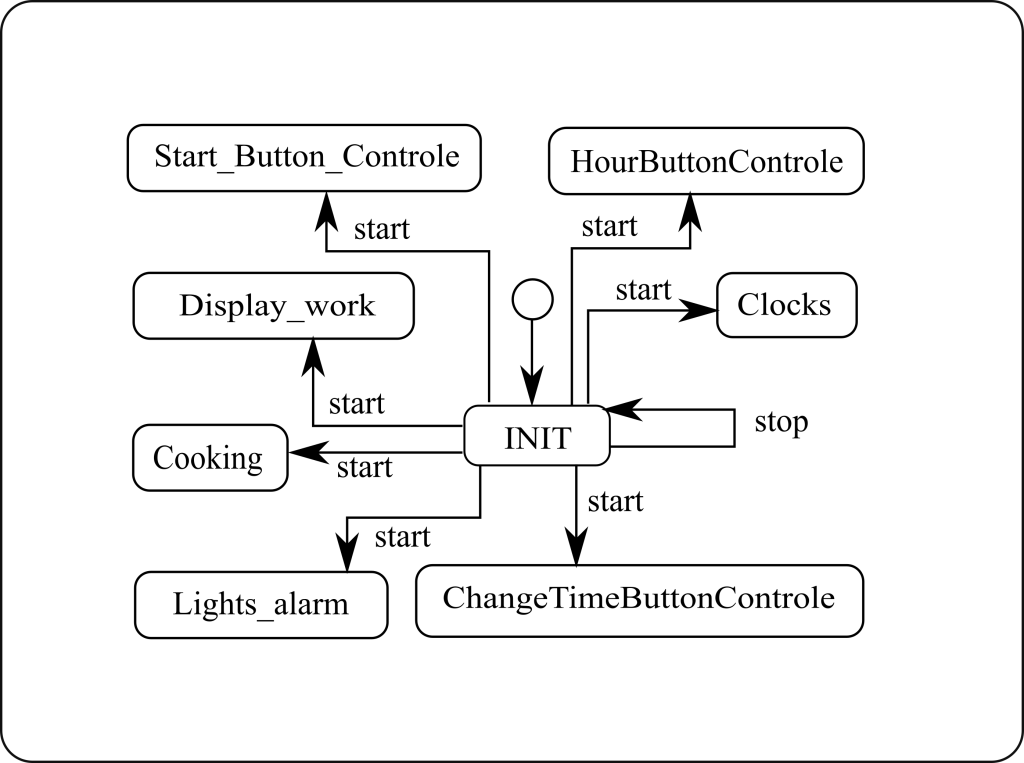


Рисунок 3 - диаграмма связи процессов по управлению

## Выбор формата представления диаграмм

### Обзор средств визуализации графов.

Рассмотрим средства визуализации графов.

1. GraphViz [28][33][37][38][39][40][69] - программное обеспечение визуализации графов с открытым исходным кодом. Пакет состоит из набора утилит командной строки и программ с графическим интерфейсом, способных обрабатывать файлы на языке DOT, а также виджетов и библиотек, облегчающих создание графов и программ для их построения, включая следующие:

- dot — инструмент создания многоуровневого графа с возможностью вывода изображения результирующего графа во множестве форматов (PNG, PDF, PostScript, SVG и ряда других). Dot располагает ребра в одном направлении (сверху вниз или слева на право), пытается избежать пересечений ребер и уменьшает их длины.

- neato — инструмент создания графа на основе «пружинной» модели («spring model», «energy minimised»). Предлагается основной для небольших графов (около 100 узлов), о структуре которых ничего не известно.

- twopi — инструмент создания графа на основе «радиальной» модели. Расположение узлов на концентрических окружностях зависит от их дистанции от данного корневого узла. Положение корневого узла можно выбрать или предоставить это программе.

- circo — инструмент создания графа на основе «круговой» модели.

- fdp — инструмент создания ненаправленного графа на основе «fdp»-модели.

- dotty — графический интерфейс для создания графов.

- lefty — программируемый графический виджет (на языке EZ[3][4]).

- Sfdp - мультимасштабная версия fdp для верстки больших графов.

GraphViz имеет множество опций для индивидуализации диаграмм. Среди них: выбор цветов, шрифтов, изменение стилей линий и узлов, создание гиперссылок и т.д. Имеет 55 видов выходных форматов файлов. Есть API для взаимодействия со многими языками, такими, как Java и Python. Форма узлов может быть различной. dotty предоставляет возможность перемещать узлы, но при этом затираются элементы, лежащие на пути перетаскивания узла.

1. Gephi [28][34] - это программный пакет сетевого анализа и визуализации с открытым исходным кодом. В нем есть инструменты раскраски вершин и рёбер по их свойствам, настройка подписей, размеров и прочих параметров. Есть экспорт в svg, png, pdf. Поддерживает импорт из БД (SQLite, MySQL, PostgreSQL и Teradata). Gephi реализует 12 алгоритмов компоновки графа. Импортирует такие форматы представления графов, как GEXF, GDF, GML, GraphML, Pajek NET, DOT, CSV, UCINET DL, TPL, Netdraw VNA, Spreadsheet. Есть API. Поддерживает только круглые узлы. Реализует возможность перетаскивать узлы.
2. Igraph [28][41][42] - это библиотека с открытым исходным кодом для создания и управления графами и анализа сетей. Она написана на C, а также существует в виде пакетов Python и R. Находится в свободном доступе под лицензией GNU General Public License Version 2. Метки узлов в графе могут быть любыми. Граф может быть представлен в форматах Pickle, Pajek, NET, NCOL, GraphML, GraphMLz, GML, Edgelist, Edges, Dimacs, Adjacency, LGL. LEDA, DOT (только запись). Реализует 13 видов укладки графа. Поддерживает PostScript, PDF, XFig, SVG, JPG, PNG.

Реализуются следующие алгоритмы расположения графов:

* circle - распределяет вершины равномерно по кругу.
* drl - уже упомянутый алгоритм для больших графов.
* fr - силовой алгоритм Fruchterman-Reingold.
* graphort - алгоритм на основе физической модели, разработанный Michael Schmuhl.
* gfr - модификация алгоритма Fruchterman-Reingold для больших графов.
* kk - алгоритм Kamada-Kawai.
* lgl - адгоритм LGL.
* random - равномерное распределение точек.
* rt - алгоритм Reingold-Tilford.
* rt\_circular - модификация предыдущего алгоритма для размещения дерева по кругу.

1. Graphistry [28][43] Реализует только одну укладку. Есть ограничение 800 тысяч на максимальное число вершин или рёбер. Распространяется под лицензией Apache License 2.0. Вычисления проводятся на сервере, их стоимость 10$ в час
2. OGDF [30][44][45][46] - это автономная библиотека C++ для графических алгоритмов, в частности для автоматического рисования графов. Библиотека доступна под лицензией GNU General Public License. Реализует hierarchical layout, hierarchical layout with predefined layering, energy-based layout, orthogonal layout. Поддерживает XML, GML, SVG форматы. Не занимается непосредственно отрисовкой графов.
3. ZGRViewer [47][48] - это графический визуализатор, реализованный на Java и основанный на масштабируемой машине визуального преобразования. Он специально предназначен для отображения графиков на DOT. Выполняет построение SVG файла с помощью GraphViz.
4. yEd [31]. Распространяется бесплатно под лицензией yEd Software License Agreement, предполагающей следующее: «Лицензиату предоставляется неисключительное и непередаваемое право на установку одной копии программного обеспечения и использование его в качестве приложения. Программное обеспечение не может использоваться как часть автоматизированного процесса» [32]. Поддерживает следующие форматы представления графов: GraphML, YGF, GML, XGML, XLS, XLSX, TGF, GED, XML + XSL, Edgelist, Nodelist, экспорт в форматы PNG, JPG, SVG, PDF, SWF, GIF, BMP, EMF, EPS, HTML. Реализует более 24 видов алгоритмов укладки графов (таких, как tree layout, radial layout и другие).
5. NetworkX [35][36][49] - библиотека для манипулирования графами под Python, распространяется свободно под BSD-new license. Узлы могут быть любыми (например, текстом, изображениями, записями XML). Ребра могут содержать произвольные данные (например, веса, временные ряды). Поддерживаются следующие форматы представления графов: GML, GraphML, Edge list, GIS Shapefile, Pajek, Sparse6, Graph6, YAML, LEDA, JSON, Pickle, GEXF, Adjacency List. Алгоритмы укладки графов предоставляются GraphViz.
6. Tulip [59][60][61] – библиотека, распространяется бесплатно под лицензией [43] GNU LESSER GENERAL PUBLIC LICENSE. Поддерживает Basic, Force Directed, Hierarchical, Misc, Multilevel, Planar, Tree layout типы. Плагины используют OGDF.
7. LEDA [58][65][66] – платная библиотека для C++ и Java, содержащая также модуль GraphWin.
8. NetDraw [62][63] – бесплатное приложение. Программа позволяет создавать структуру с оптимальным расположением узлов, трансформировать граф, перетаскивать отдельные узлы вручную. Сохраняет в форматы JPEG, BITMAP, Metafile.

Более наглядное сравнение представлено ниже (см. Таблицу 4). Зеленым цветом выделены критерии, которые удовлетворяют озвученным ранее критериям, красным – которые не удовлетворяют. Также указано количество реализуемых алгоритмов укладки графа [29] и приведен список поддерживаемых форматов хранения графов.

Таблица 4 - обзор средств визуализации графов

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Название** | **Платно для коммерческого использования** | **Возможность перетаскивания** | **Auto layout** | **Количество алгоритмов укладки графа** | **Формат представления графов** | **Количество поддерживаемых форматов** | **Настройка параметров визуализации** | **Наличие API** |
| GraphViz | - | +\* | + | 11 | dot (gv) | 55 | + | + |
| Gephi | - | + | + | 12 | GEXF, GDF, GML, GraphML, Pajek NET, gv, CSV, UCINET DL, Tulip TPL, Netdraw VNA, Spreadsheet | 9 | + | + |
| Igraph | - | - | + | 13 | Pickle, pajek, net, ncol, GraphMLz, GraphML, gml, edgelist, edges, edge, dl(только чтение), dimacs, adjacency, lgl. | 6 | + | + |
|  |  |  |  |  | Leda, graphviz, dot (только запись) |  |  |  |
| 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 | + | ? |

### Обзор форматов хранения графов.

Список наиболее распространенных форматов хранения графов:

- GraphML [50][51][52]

- GV [53][54]

- XGML [55]

- GML [56]

- Node list [62]

- Edge list [62]

- Pajek [57]

- LEDA [58][65][66]

- TLP [59][60][61]

- GW [58][65][66]

- GEXF [67][68]

Краткий обзор наиболее распространенных форматов хранения графов приведен в таблице 5. Схема возможных преобразований форматов представлены на диаграмме (см. Рис 4).

Таблица 5 - обзор форматов хранения графов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Название** | **Хранение координат** | **Форма вершин как в нотациях** | **Ориентированный граф** | **Подписи над ребрами** | **Поддерживающие средства** |
| GraphML | + | + | + | + | Gephi, Igraph, OGDF, Yed, NetworkX |
| Gv | - | + | + | + | GraphViz, Gephi, Igraph\*, OGDF, ZGRViewer, NetworkX |
| Xgml\* | + | + | + | + | Yed |
| Gml | + | + | + | + | Gephi, Igraph, OGDF, Yed, NetworkX, Tulip, LEDA |
| Node list | - | - | + | - | Yed, NetDraw |
| Edge list | - | - | + | - | OGDF, Yed, NetworkX, NetDraw |
| Pajek | + | + | + | + | Gephi, Igraph, NetworkX, NetDraw |
| Leda | + | + | + | + | Igraph, OGDF, NetworkX, LEDA |
| TLP | + | + | + | + | OGDF, Gephi, Tulip |
| Gw | + | + | + | + | LEDA |
| GEXF | + | + | + | - | Gephi, OGDF, NetworkX, Tulip |

Красным помечены не удовлетворяющие требованиям критерии, зеленым – удовлетворяющие. При анализе рассматривались такие свойства форматов, как возможность хранения координат, чтобы после автоматической укладки графа можно было сохранить и закрыть файл с диаграммой. При повторном открытии файла укладка должна сохраниться (также как и модификации, сделанные пользователем). Кроме того, рассматривалась возможность создания диаграмм, удовлетворяющих созданным ранее графическим нотациям (например, возможность указывать подписи над ребрами, задавать необходимую форму вершин и т.д.).

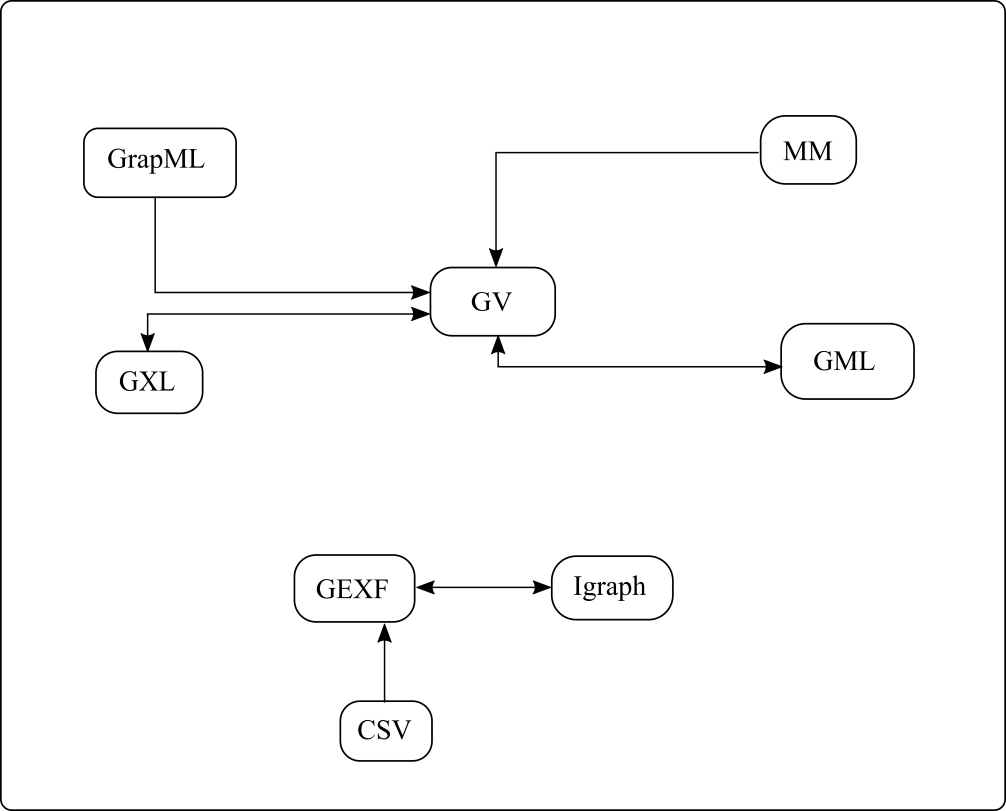


Рисунок 4 - схема возможных конвертаций графов

В результате анализа были выделены два языка описания графов: GML и GraphML. Они оказались наиболее поддерживаемыми форматами, удовлетворяющими поставленным критериям. Также при отборе учитывалось количество форматов, в которые можно перевести рассматриваемый формат.

### Сравнение GML и GraphML.

Для окончательного выбора одного из форматов рассмотрим более подробно сравнение форматов GML и GraphML (см. Таблицу 6).

Таблица 6 - сравнение GML и GraphML

|  |  |  |
| --- | --- | --- |
| Критерий сравнения | GML | GraphML |
| Размер файла | x | 2x |
| Легок для восприятия человеком | + | - |
| Возможность хранения ссылок в вершинах | - | + |

### Выводы.

По результатам анализа было решено генерировать диаграмму состояний процесса в формате GML, так как так такая диаграмма будет создаваться для каждого процесса, которых может быть много. Следовательно, будет лучше, если файлы диаграмм будут иметь небольшой размер.

Остальные два вида диаграмм будут генерироваться в формате GraphML, так как он позволяет хранить ссылки внутри вершин. В каждой вершине процесса будет помещена ссылка на диаграмму состояний этого процесса.

Такое решение позволит не только сэкономить память, но и обеспечить удобную навигацию по структуре проекта.

# ГЛАВА 3

## Архитектура модуля

При проектировании архитектуры разрабатываемого модуля использовался шаблон проектирования Model–view–controller (MVC) с активной Моделью. Такое решение позволяет уменьшить время, необходимое для внесения изменений в отдельные части модуля по сравнению с монолитной архитектурой. Схема взаимодействия частей модуля в рамках модели MVC представлена на рис. 5.

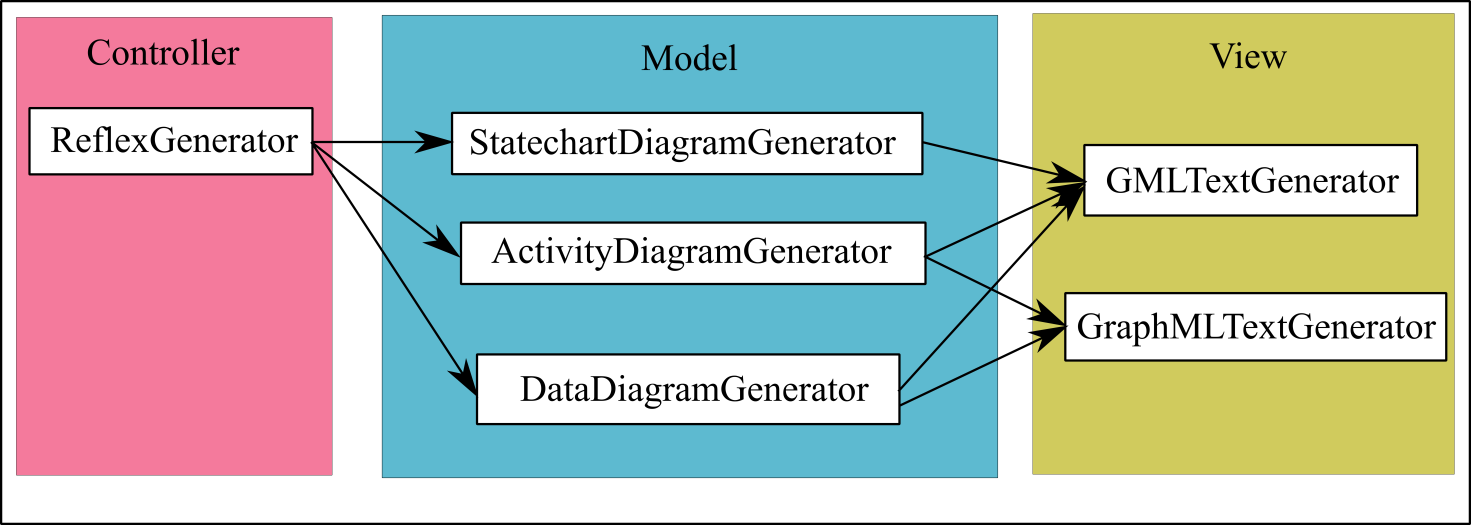


Рисунок 5 - Схема взаимодействия частей модуля в рамках модели MVC

Сначала контроллер проходит по списку процессов и запускает создание модели диаграммы состояний текущего процесса по абстрактному синтаксическому дереву программы, потом запускает генератор текста диаграммы в формате GML.

После этого контроллер запускает генерацию модели диаграмм связи процессов по данным и по управлению на основе абстрактного синтаксического дерева, полученного модулем на вход. Далее он передает полученные модели в модуль разделения диаграмм, который разбивает модель на независимые части (то есть, если в одной модели оказалось две диаграммы, не имеющие общих вершин, они будут разделены на две разные модели).

После получения списка разделенных на независимые части моделей контроллер запускает генерацию текста диаграмм в формате GraphML, при этом в каждую вершину процесса помещается ссылка на диаграмму состояний этого процесса, сгенерированную ранее.

## Реализация модуля

### Варианты реализации модуля

В качестве возможных реализаций рассматривались следующие варианты:

- Встраивание модуля в существующий устаревший транслятор языка Рефлекс.

- Создание независимого приложения, данные которому будут подаваться новой разрабатываемой IDE для языка Рефлекс.

- Создание Eclipse-плагина с использованием технологий Xtext и Xtend.

Реализацию модуля решено выполнить в виде Eclipse-плагина с использованием технологий Xtext и Xtend для написания генератора по следующим причинам:

* Простота интеграции в новую IDE языка Reflex.
* Автоматический вызов модуля из IDE.
* Использование актуальной грамматики языка.

### Реализация модуля разделения моделей диаграмм без общих вершин

#### Входные данные.

На вход модуль разделения диаграмм получает модель диаграммы процессов, то есть список ребер (ArrayList), где в каждом элементе содержится информация о том, откуда и куда идет ребро, а также о надписи над ребром. Также модель диаграммы представляется ассоциативным массивом (HashMap), где ключом является имя вершины (то есть имя процесса), а значением – описание вершины диаграммы с ее id, именем процесса, формой вершины и информацией о посещении (в ней после работы модуля разделения лежит номер компоненты, к которой вершина относится).

#### Выходные данные.

В результате работы модуля создается два массива (ArrayList), элементами первого являются списки ребер независимых диаграмм, элементами второго - ассоциативные массивы вершин соответствующих диаграмм.

#### Алгоритм.

Алгоритм работы модуля разделения диаграмм процессов, не имеющих общих вершин, описан ниже.

Не посещенные вершины процессов добавляются в очередь. Далее достаем последовательно из очереди один элемент. После этого:

* Если он еще не был посещен, то отмечаем его как посещенный и добавляем в текущую компоненту. Получаем список всех его потомков и добавляем их в очередь.
* Если он уже был посещен.

Если он относится к компоненте, отличной от текущей компоненты, то:

* + Если номер компоненты меньше текущей.

Номер текущей компоненты заменяется на номер найденной компоненты, все вершины текущей компоненты добавляются к вершинам найденной.

* + Если номер компоненты больше текущей.

Номер текущей компоненты остается тем же, все элементы найденной компоненты добавляются к текущей компоненте.

Блок-схема алгоритма представлена на рис. 6.

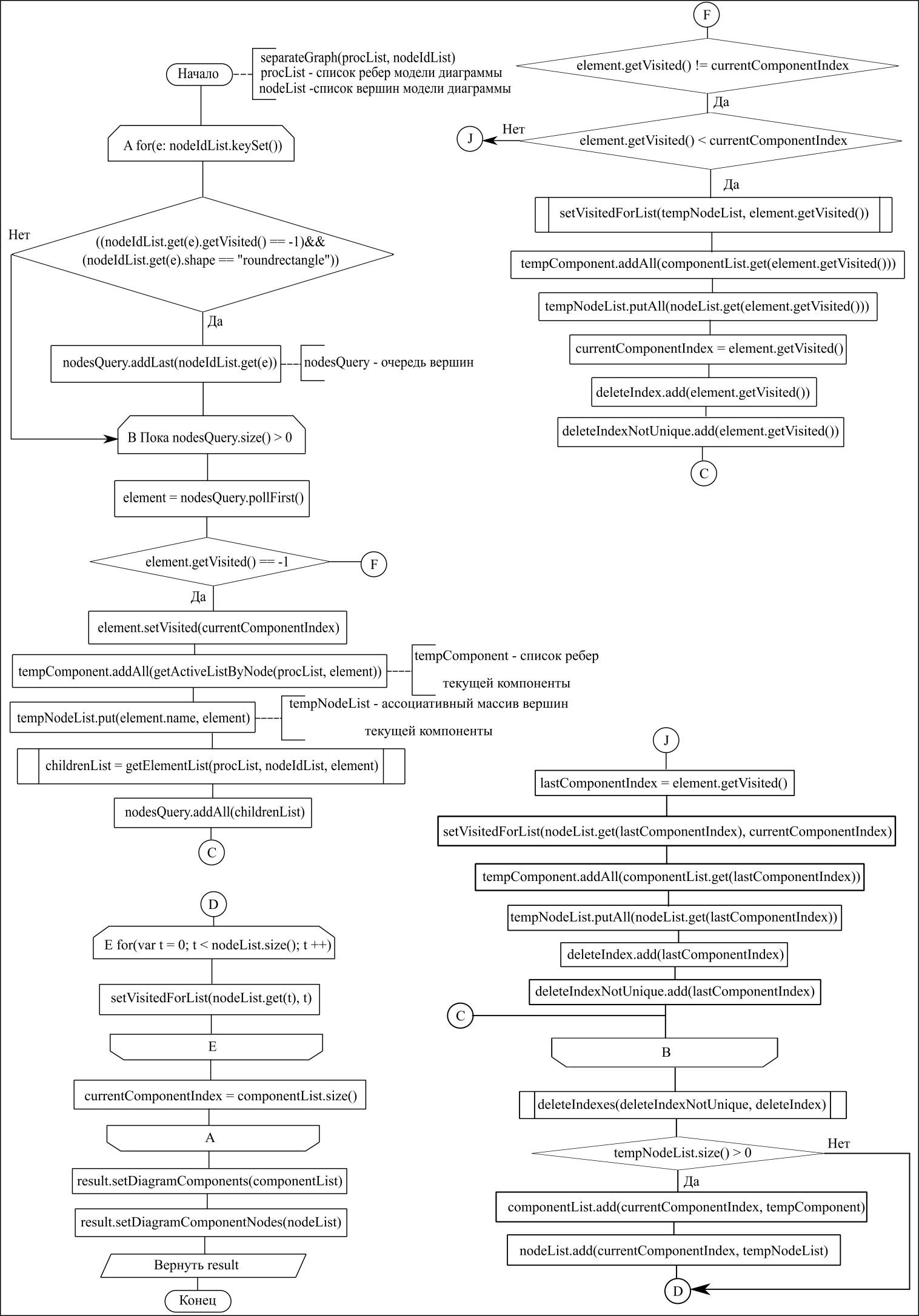


Рисунок 6 - Блок-схема алгоритма разделения несвязанных диаграмм

## Алгоритм использования модуля

Схема алгоритма использования разработанного модуля представлена на рис. 7. После открытия Рефлекс IDE и написания кода, можно нажать кнопку сохранения файла, после чего автоматически сгенерируются диаграммы связи процессов по данным, по управлению и диаграммы состояний для каждого процесса. Открывать диаграммы рекомендуется одним из средств визуализации графов, наиболее подходящим при этом, по результатам анализа, оказался yEd, поддерживающий автоматическую укладку графа на плоскость, а также возможность редактирования диаграммы.

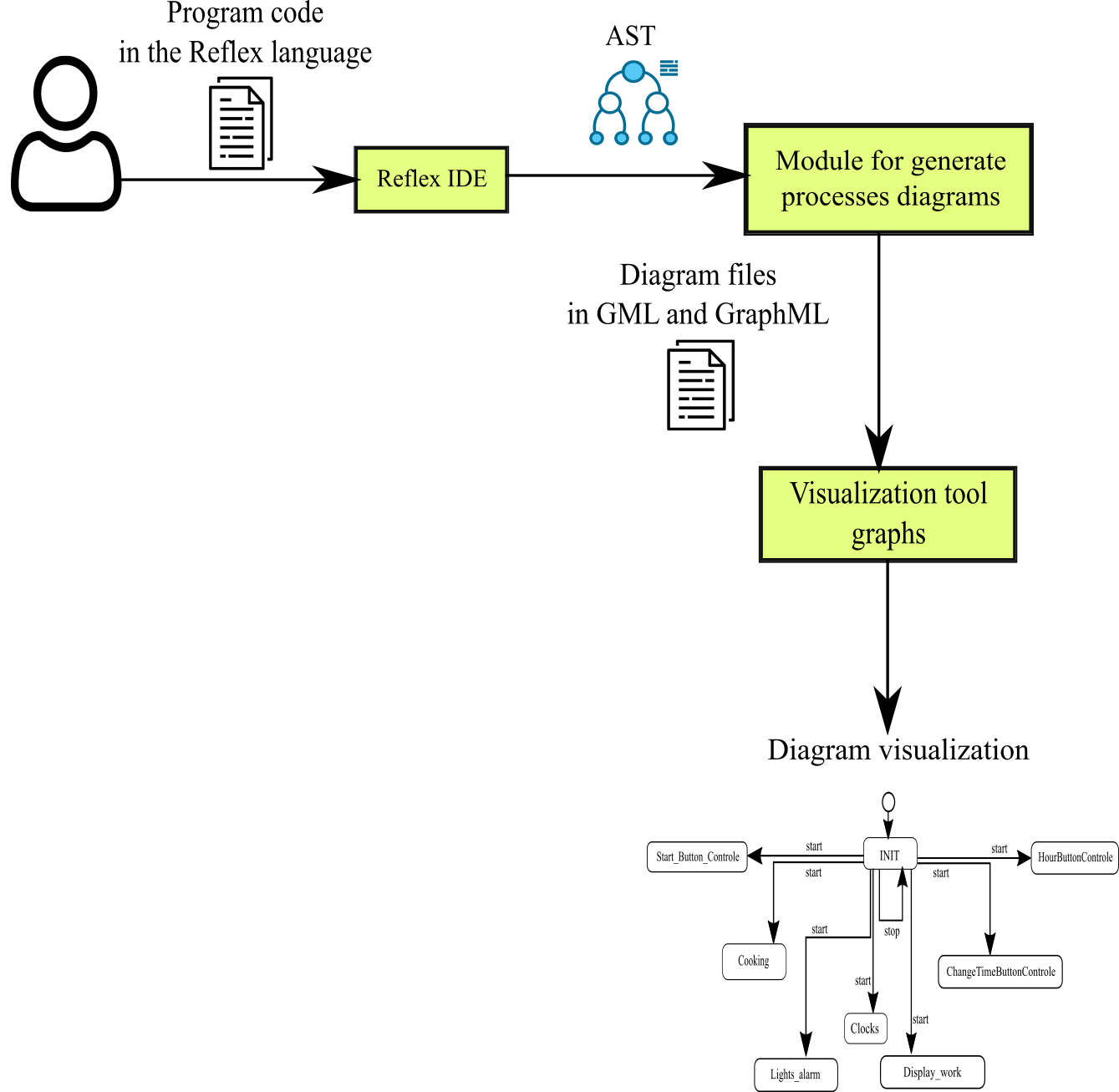


Рисунок 7 - схема алгоритма использования модуля визуализации диаграмм процессов

# ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы были выполнены следующие задачи:

1. проведен анализ:
   1. специфики ПОП на языке Reflex с точки зрения необходимости визуализации диаграмм;
   2. средств визуализации диаграмм по коду для языков общего назначения и для специализированных языков;
2. спроектирована система:
   1. сформулированы требования к создаваемому программному модулю;
   2. разработаны диаграммы для отображения связей процессов;
   3. определен формат представления диаграмм;
3. разработана архитектура модуля, удовлетворяющая обозначенным требованиям;
4. реализован модуль визуализации;
5. проведено тестирование созданной реализации, апробация ее на практике.
6. сделан доклад на Международной научной студенческой конференции. В сборнике материалов конференции опубликованы тезисы (см. приложение Б).
7. создано руководство оператора (см. приложение В).
8. создано описание программы (см. приложение Г).

В результате работы для языка Рефлекс был создан модуль генерации диаграмм состояний процессов, диаграмм связи процессов по данным и по управлению, который автоматизирует создание графической документации и упрощает поддержку проектов на языке Рефлекс.

Практическая апробация показала, что использование модуля на несколько порядков сокращает время создания диаграмм (с нескольких часов до пары секунд) и гарантирует отсутствие ошибок, вызванных человеческим фактором.

Выпускная квалификационная работа выполнена мной самостоятельно и с соблюдением правил профессиональной этики. Все использованные в работе материалы и заимствованные принципиальные положения (концепции) из опубликованной научной литературы и других источников имеют ссылки на них. Я несу ответственность за приведенные данные и сделанные выводы.

Я ознакомлен с программой государственной итоговой аттестации, согласно которой обнаружение плагиата, фальсификации данных и ложного цитирования является основанием для не допуска к защите выпускной квалификационной работы и выставления оценки «неудовлетворительно».

|  |  |
| --- | --- |
| Беленькая София Евгеньевна\_ | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| ФИО студента | Подпись студента |

« \_\_\_\_ »\_\_\_\_\_\_\_\_\_\_\_2020 г.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ И ЛИТЕРАТУРЫ

1. Розов А. С., В. Е. Зюбин, Нефедов Д. В. Программирование встраиваемых микроконтроллерных систем на основе гиперпроцессов//Вестн. НГУ. Серия: Информационные технологии. 2017. Т. 15, №4. С. 64-73.
2. Зюбин В. Е. Процесс-ориентированное программирование: Учеб. пособие //Новосиб. гос. ун-т. – 2011.
3. Розов А. С. и др. Практическая апробация языка IndustrialC на примере автоматизации установки термовакуумного напыления //Вестник Новосибирского государственного университета. Серия: Информационные технологии. – 2017. – Т. 15. – №. 3.
4. Лях Т. В., Зюбин В. Е., Сизов М. М. Опыт применения языка Reﬂex при автоматизации Большого солнечного вакуумного телескопа //Промышленные АСУ и контроллеры. – 2016. – №. 7. – С. 37-43.
5. Зюбин В. Е. Язык Рефлекс. Математическая модель алгоритмов управления //Датчики и системы. – 2006. – №. 5. – С. 24-30.
6. В. Е. Зюбин «Си с процессами»: язык программирования логических контроллеров //Мехатроника. 2006. № 12. C. 31–35
7. В. Е. Зюбин Язык «Рефлекс» – диалект Си для программируемых логических контроллеров // Шестая международная научно-практическая конференция "Средства и системы автоматизации" CSAF’06 // Томск, 1-3 ноября 2005 г. Томск: ТУСУР, 2005
8. TOP IDE index [Электронный ресурс] : The Top IDE Index is created by analyzing how often IDEs' download page are searched on Google / Pierre Carbonnelle, под лицензией Creative Commons Attribution 3.0 Unported License. - Электрон. граф. и текстовые дан. - Режим доступа : http://pypl.github.io/IDE.html. - Англ. - Загл. со страницы Интернета. - Обновляется ежегодно.
9. Практическое руководство. Добавление диаграмм классов в проекты [Электронный ресурс] : документация по Visual Studio / Microsoft 2020. - Электрон. граф. и текстовые дан. - Режим доступа : https://docs.microsoft.com/ru-ru/visualstudio/ide/class-designer/how-to-add-class-diagrams-to-projects?view=vs-2019. - Загл. с домашней страницы Интернета.
10. The Best UML Diagramming Software Available [Электронный ресурс] : astah / ChangeVision 2020. - Электрон. граф. и текстовые дан. - Режим доступа : https://astah.net/products/astah-uml/. - Загл. со страницы Интернета.
11. MagicDraw [Электронный ресурс] / No Magic 2020. - Электрон. граф. и текстовые дан. - Режим доступа : https://www.nomagic.com/products/magicdraw. - Англ. - Загл. со страницы Интернета.
12. Agile CASE tool for software design & analysis [Электронный ресурс] / Dušan Rodina. - Электрон. граф. и текстовые дан. - Режим доступа : https://www.softwareideas.net. - Англ. - Загл. со страницы Интернета.
13. BOUML [Электронный ресурс] / Bruno Pagès. - Электрон. граф. и текстовые дан. - Режим доступа : https://www.bouml.fr. - Англ. - Загл. со страницы Интернета.
14. Visual Paradigm [Электронный ресурс] / Visual Paradigm. - Электрон. граф. и текстовые дан. - Режим доступа : https://www.visual-paradigm.com. - Англ. - Загл. со страницы Интернета.
15. Rational Rose [Электронный ресурс] / KPMS. - Электрон. граф. и текстовые дан. - Режим доступа : https://www.kpms.ru/Automatization/Rational\_Rose.htm. - Загл. со страницы Интернета.
16. Enterprise Architect [Электронный ресурс] : Make Your Vision a Reality / Sparx Systems Pty Ltd. - Электрон. граф. и текстовые дан. - Режим доступа : https://sparxsystems.com. - Англ. - Загл. со страницы Интернета.
17. IntelliJ IDEA [Электронный ресурс] / JetBrains s.r.o. - Электрон. граф. и текстовые дан. - Режим доступа : https://www.jetbrains.com/idea/. - Англ. - Загл. со страницы Интернета.
18. PowerDesigner [Электронный ресурс] / ООО "Сайбейс". - Электрон. граф. и текстовые дан. - Режим доступа : https://www.sybase.ru/products/powerdesigner. - Загл. со страницы Интернета.
19. Apache NetBeans 11.3 [Электронный ресурс] / The Apache Software Foundation. - Электрон. граф. и текстовые дан. - Режим доступа : https://netbeans.apache.org. - Загл. со страницы Интернета.
20. LabVIEW [Электронный ресурс] / National Instruments Russia. - Электрон. граф. и текстовые дан. - Режим доступа : http://www.labview.ru. - Загл. со страницы Интернета.
21. NClass [Электронный ресурс] / Balazs Tihanyi. - Электрон. граф. и текстовые дан. - Режим доступа : http://nclass.sourceforge.net/index.html. - Загл. с домашней страницы Интернета.
22. UML Software Modeling Tool [Электронный ресурс] / Altova. - Электрон. граф. и текстовые дан. - Режим доступа : https://www.altova.com/umodel. - Загл. с домашней страницы Интернета.
23. Интегрированные среды разработки программного обеспечения для микроконтроллеров [Электронный ресурс] : Портал магистров : Индивидуальный раздел / С. В. Стойчев. - Электрон. граф. и текстовые дан. - Режим доступа : http://masters.donntu.org/2014/etf/stoychev/ind/index.htm. - Загл. с экрана.
24. Официальный сайт AXEL [Электронный ресурс] : LogicLab / 2017 AXEL S.R.L - SOFTWARE FOR INDUSTRIAL AUTOMATION. - Электрон. граф. и текстовые дан. - Режим доступа : https://www.axelsoftware.it/en/. – Англ. - Загл. с домашней страницы Интернета.
25. Harel D. Statecharts: A visual formalism for complex systems //Science of computer programming. – 1987. – Т. 8. – №. 3. – С. 231-274.
26. UML O. M. G. OMG Unified Modeling Language (UML) Specification: Infrastructure, Version 2.2, Final Available Specification. – 2009.
27. Drusinsky D. Modeling and verification using UML statecharts: a working guide to reactive system design //Runtime Monitoring and Execution-based Model Checking, Newnes. – 2006.
28. Хабр [Электронный ресурс] : Визуализация больших графов для самых маленьких / Open Data Science. - Электрон. прогр., граф. и текстовые дан. - Режим доступа : https://habr.com/ru/company/ods/blog/464715/. - Загл. со страницы Интернета.
29. Gibson H., Faith J., Vickers P. A survey of two-dimensional graph layout techniques for information visualisation //Information visualization. – 2013. – Т. 12. – №. 3-4. – С. 324-357.
30. M. Chimani, C. Gutwenger, M. Jünger, G. W. Klau, K. Klein, P. Mutzel. The Open Graph Drawing Framework (OGDF). Chapter 17 in: R. Tamassia (ed.), Handbook of Graph Drawing and Visualization, CRC Press, 2014.
31. Официальный сайт yWorks [Электронный ресурс] : yEd Graph Editor / yWorks. - Электрон. прогр., граф. и текстовые дан. - Режим доступа : https://www.yworks.com/products/yed. - Англ. - Загл. с домашней страницы Интернета.
32. yEd Software License Agreement [Электронный ресурс] / yWorks. - Version 1.2. - Электрон. текстовые дан. - Режим доступа : https://www.yworks.com/resources/yed/license.html. - Англ. - Загл. с экрана.
33. Graphviz - Graph Visualization Software [Электронный ресурс] : Welcome to Graphviz. - Электрон. прогр., граф. и текстовые дан. - Режим доступа : http://www.graphviz.org. - Англ. - Загл. с домашней страницы Интернета.
34. Gephi [Электронный ресурс] : The Open Graph Viz Platform / Gephi. - Электрон. прогр., граф. и текстовые дан. - Режим доступа : https://gephi.org. - Англ. - Загл. с домашней страницы Интернета.
35. Ещё один ИТ-шный бложек [Электронный ресурс] : NetworkX [заметки о визуализации графов на python] / пользователь esemi Технологии Blogger. - Электрон. граф. и текстовые дан. - Систем. требования: язык программирования Python. - Режим доступа : http://blog.esemi.ru/2011/12/networkx-python.html. - Загл. со страницы Интернета.
36. NetworkX [Электронный ресурс] : Software for complex networks / NetworkX developers. - Электрон. прогр., граф. и текстовые дан. - Систем. требования: язык программирования Python. - Режим доступа : https://networkx.github.io. - Англ. - Загл. с домашней страницы Интернета.
37. Хабр [Электронный ресурс] : Использование Graphviz для построения блок-схем / Разработчик Андрей @kspksp. - Электрон. граф. и текстовые дан. - Режим доступа : https://habr.com/ru/post/337078/. - Загл. со страницы Интернета.
38. LinuxShare [Электронный ресурс] : Введение в GraphViz / В. Г. Вислобоков. - Электрон. граф. и текстовые дан. - Режим доступа : http://linuxshare.ru/docs/misc/graphviz.html. - Загл. с домашней страницы Интернета.
39. Ещегодник [Электронный ресурс] = Информационно-образовательный сайт для студентов, аспирантов и коллег Тушавина В. А., созданный и пополняемый им самим безвозмездно : Graphviz – рисуем графы / В. А. Тушавин. - Электрон. граф. и текстовые дан. - Режим доступа : https://tushavin.ru/graphviz/. - Загл. с домашней страницы Интернета.
40. GraphViz Reference [Электронный ресурс]. - Электрон. граф. и текстовые дан. - Режим доступа : http://www.graphviz.org/doc/info/. - Англ. - Загл. со страницы Интернета.
41. Pyton-igraph Manual [Электронный ресурс] : Tutorial / The igraph core team, Code licensed under GNU GPL 2 or later, documentation under GNU FDL. - Электрон. граф. и текстовые дан. - Систем. требования: язык программирования Python. - Режим доступа : https://igraph.org/python/doc/tutorial/tutorial.html. - Англ. - Загл. со страницы Интернета.
42. Package ‘igraph’ [Электронный ресурс] : Network Analysis and Visualization / Maintainer Gabor Csardi. - Version 1.2.4.1. - Электрон. граф. и текстовые дан. (413 с.). - 2019-04-22 06:57:32 UTC. - Режим доступа : https://igraph.org/r/doc/igraph.pdf. - Англ. - Загл. с электрон. документа.
43. Graphistry [Электронный ресурс] : Get Started / Graphistry. - Электрон. прогр. и текстовые дан. - Режим доступа : https://www.graphistry.com/get-started. - Англ. - Загл. со страницы Интернета.
44. OGDF [Электронный ресурс] = Open Graph Drawing Framework / Osnabrück University, Theoretical Computer Science, Germany; TU Dortmund, Chair XI Algorithm Engineering, Germany; University of Konstanz, Life Science Informatics, Germany; University of Cologne, Chair Jünger, Germany; TU Ilmenau, Discrete Mathematics and Algebra, Germany; Markus Chimani, project manager; Karsten Klein, project manager; Stephan Beyer, release manager; Martin Gronemann; Ivo Hedtke; Denis Kurz; Samir Mahmalat; Jöran Schierbaum; Jens M. Schmidt; Sebastian Semper; Mirko H. Wagner; Tilo Wiedera; Bernd Zey. - Электрон. прогр. и текстовые дан. - Систем. требования: язык программирования C++. - Режим доступа : https://ogdf.uos.de. - Англ. - Загл. с домашней страницы Интернета.
45. Chimani M. et al. The Open Graph Drawing Framework (OGDF) //Handbook of graph drawing and visualization. – 2013. – Т. 2011. – С. 543-569.
46. OGDF [Электронный ресурс] : This is a release mirror for OGDF, the Open Graph Drawing Framework/Open Graph algorithms and Data structure Framework. / Osnabrück University, Theoretical Computer Science, Germany; TU Dortmund, Chair XI Algorithm Engineering, Germany; University of Konstanz, Life Science Informatics, Germany; University of Cologne, Chair Jünger, Germany; TU Ilmenau, Discrete Mathematics and Algebra, Germany; Markus Chimani, project manager; Karsten Klein, project manager; Stephan Beyer, release manager; Martin Gronemann; Ivo Hedtke; Denis Kurz; Samir Mahmalat; Jöran Schierbaum; Jens M. Schmidt; Sebastian Semper; Mirko H. Wagner; Tilo Wiedera; Bernd Zey. - Электрон. прогр. и текстовые дан. - Систем. требования: язык программирования C++. - Режим доступа : https://github.com/ogdf/ogdf. - Англ. - Загл. с домашней страницы Интернета.
47. ZGRViewer, a GraphViz/DOT Viewer [Электронный ресурс] / E. Pietriga. - Электрон. прогр. и текстовые дан. - Режим доступа : http://zvtm.sourceforge.net/zgrviewer.html
48. Pietriga E. A toolkit for addressing hci issues in visual language environments //2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05). – IEEE, 2005. – С. 145-152.
49. Aric A. Hagberg, Daniel A. Schult and Pieter J. Swart, “Exploring network structure, dynamics, and function using NetworkX”, in Proceedings of the 7th Python in Science Conference (SciPy2008), Gäel Varoquaux, Travis Vaught, and Jarrod Millman (Eds), (Pasadena, CA USA), pp. 11–15, Aug 2008.
50. The GraphML File Format [Электронный ресурс] / GraphML Team. - Электрон. текстовые дан. - Режим доступа : http://graphml.graphdrawing.org. - Англ. - Загл. с домашней страницы Интернета. – Послед. обновление 24-Jan-2019 19:22:23 CET.
51. CIT Forum [Электронный ресурс] : GraphML - язык описания графов / В. П. Шокоров. - Электрон. текстовые дан. - Режим доступа : http://citforum.ru/internet/xml/graphml/. – Пер. Электронного ресурса The GraphML File Format. - Загл. с домашней страницы Интернета.
52. В.Н. Касьянов, ЯЗЫК ПРЕДСТАВЛЕНИЯ ГРАФОВ GRAPHML: БАЗОВЫЕ СРЕДСТВА1 [Электронный ресурс] : При частичной финансовой поддержке Российского фонда фундаментальных исследований (грант РФФИ ) / В.Н. Касьянов. - Электрон. текстовые дан. - Режим доступа : https://docplayer.ru/31018473-V-n-kasyanov-yazyk-predstavleniya-grafov-graphml-bazovye-sredstva-1.html. - Загл. с электрон. документа.
53. DOT\_(язык) [Электронный ресурс] : Википедия / Wikimedia Foundation, Inc. - Электрон. граф. и текстовые дан. - Режим доступа : https://ru.wikipedia.org/wiki/DOT\_(язык) . - Загл. со страницы Интернета.
54. The DOT Language [Электронный ресурс] - Электрон. текстовые дан. - Режим доступа : https://graphviz.gitlab.io/\_pages/doc/info/lang.html. - Англ. - Загл. со страницы Интернета.
55. yFiles for Java Developer's Guide [Электронный ресурс] : XGML : Chapter 9. Input and Output / yWorks GmbH. - Электрон. граф. и текстовые дан. - Режим доступа : https://docs.yworks.com/yfiles/doc/developers-guide/xgml.html. - Англ. - Загл. с домашней страницы Интернета.
56. yFiles for Java Developer's Guide [Электронный ресурс] : GML : Chapter 9. Input and Output / yWorks GmbH. - Электрон. граф. и текстовые дан. - Режим доступа : https://docs.yworks.com/yfiles/doc/developers-guide/gml.html. - Англ. - Загл. с домашней страницы Интернета.
57. Pajek: Exporting pictures from Pajek to EPS [Электронный ресурс] / Vladimir Batagelj, Andrej Mrvar. - Электрон. граф. и текстовые дан. - Режим доступа : http://vlado.fmf.uni-lj.si/pub/networks/pajek/doc/draweps.htm. - Англ. - Загл. с домашней страницы Интернета.
58. Native File Format for Graphs [Электронный ресурс] / Algorithmic Solutions Software GmbH. - Электрон. прогр. и текстовые дан. - Режим доступа : http://www.algorithmic-solutions.info/leda\_guide/graphs/leda\_native\_graph\_fileformat.html. - Англ. - Загл. со страницы Интернета.
59. Tulip [Электронный ресурс] : Tulip software graph format (TLP) / David Auber; Original Author, Patrick Mary. - Электрон. прогр. и текстовые дан. - Режим доступа : https://tulip.labri.fr/TulipDrupal/?q=tlp-file-format. - Англ. - Загл. с домашней страницы Интернета.
60. Tulip [Электронный ресурс] : LGPL: License / David Auber; Original Author, Patrick Mary. - Электрон. текстовые дан. - Режим доступа : https://tulip.labri.fr/TulipDrupal/?q=licence. - Англ. - Загл. со страницы Интернета.
61. Tulip [Электронный ресурс] : Tulip Screenshots / David Auber; Original Author, Patrick Mary. - Электрон. граф. дан. - Режим доступа : https://tulip.labri.fr/TulipDrupal/?q=screenshots. - Англ. - Загл. со страницы Интернета.
62. Компьютерная визуализация социальных сетей [Электронный ресурс] / Александр Прохоров, Николай Ларичев. - Электрон. текстовые и граф. дан. - Режим доступа : https://compress.ru/article.aspx?id=16593#NetDraw. - Загл. со страницы Интернета.
63. Borgatti, S.P., 2002. NetDraw Software for Network Visualization. Analytic Technologies: Lexington, KY
64. Algorithmic solutions software GMBH [Электронный ресурс] : LEDA Graphs for Java / Algorithmic Solutions Software GmbH. - Электрон. прогр. и текстовые дан. - Режим доступа : https://algorithmic-solutions.com/index.php/products/leda-graphs-for-java. - Англ. - Загл. с домашней страницы Интернета.
65. Algorithmic solutions software GMBH [Электронный ресурс] : LEDA Graphs for C++ / Algorithmic Solutions Software GmbH. - Электрон. прогр. и текстовые дан. - Режим доступа : https://algorithmic-solutions.com/index.php/products/leda-for-c. - Англ. - Загл. с домашней страницы Интернета.
66. GEXF File Format [Электронный ресурс] / Sebastien Heymann; Gephi Consortium, Mathieu Bastian; Gephi Consortium, Mathieu Jacomy; SciencesPo MediaLab, Camille Maussang; Linkfluence SAS, Antonin Rohmer; Linkfluence SAS, Julian Bilcke; ISC-PIF, Alexis Jacomy; ISC-PIF. - Электрон. прогр. и текстовые дан. - Режим доступа : https://gephi.org/gexf/format/schema.html. - Англ. - Загл. с домашней страницы Интернета.
67. GEXF 1.2draft Primer [Электронный ресурс] / Sebastien Heymann; Gephi Consortium, Mathieu Bastian; Gephi Consortium, Mathieu Jacomy; SciencesPo MediaLab, Camille Maussang; Linkfluence SAS, Antonin Rohmer; Linkfluence SAS, Julian Bilcke; ISC-PIF, Alexis Jacomy; ISC-PIF. - Электрон. текстовые дан. (25 c.) - Режим доступа : https://gephi.org/gexf/1.2draft/gexf-12draft-primer.pdf. - Англ. - Загл. с электрон. документа.
68. Graphviz and dynagraph – static and dynamic graph drawing tools [Электронный ресурс] / John Ellson , Emden R. Gansner , Eleftherios Koutsofios , Stephen C. North , Gordon Woodhull. – Электрон. текстовые дан. - Режим доступа : http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.96.3776. - Англ.

# Список публикаций автора по теме выпускной квалификационной работы

СПИСОК

ПУБЛИКАЦИЙ

**БЕЛЕНЬКАЯ СОФИЯ ЕВГЕНЬЕВНА**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **№**  п/п | Наименование  работы, её вид | Форма  работы | Выходные  данные | Объём  в стр. | Соавторы |
| 1 | 2 | 3 | 4 | 5 | 6 |

**научные работы:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Разработка программного модуля визуализации диаграмм процессов по спецификации на языке Reflex (тезисы) | печ. | Информационные технологии : Материалы 58-й Междунар. науч. студ. конф. 2020 г. / Новосиб. гос. ун-т. — Новосибирск : ИПЦ НГУ, 2020. — 240 с. | 1/240 |  |

Автор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ С. Е. Беленькая

(подпись)

00.00.2020

**Список верен:**

Заведующий кафедрой КТ ФИТ, профессор \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В. Е. Зюбин

(подпись)

Секретарь ученого совета ФИТ НГУ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ О. А. Кутненко

(подпись)

# ПРИЛОЖЕНИЕ А программа управления микроволновой печью

program Micro

{

const ON 1;

const OFF 0;

const OPEN 1;

const CLOSE 0;

const ONE\_SEC 100;

const ONE\_MIN 60\*ONE\_SEK;

/\*===========================================================\*/

/\*========= БАЗОВЫЕ АДРЕСА МОДУЛЕЙ УСО ===================\*/

/\*===========================================================\*/

const BA\_FPGA1\_U1 0XA110; /\* ВХОД \*/

const BA\_FPGA3\_U1 0XA910; /\* ВЫХОД \*/

/\*######## ОПИСАНИЕ РЕГИСТРОВ МОДУЛЕЙ УСО #########\*/

/\*###########################################################\*/

input LOG\_INPUT 0x01 0x02 8;

output LOG\_OUTPUT 0x03 0x04 16;

proc INIT{

/\* ВХОДНЫЕ СИГНАЛЫ: \*/

bool K\_BUTTON\_START = {LOG\_INPUT[1]} global;

bool K\_DOOR = {LOG\_INPUT[1]} global;

bool K\_BUTTON\_CHANGE\_TIME = {LOG\_INPUT[1]} global;

bool K\_BUTTON\_HOUR = {LOG\_INPUT[1]} global;

/\* ВЫХОДНЫЕ СИГНАЛЫ (т.к. привязаны к модулю выходов): \*/

bool Y\_WARM = {LOG\_OUTPUT[1]} global;

bool Y\_LAMP = {LOG\_OUTPUT[1]} global;

bool Y\_ALARM = {LOG\_OUTPUT[1]} global;

bool COOK\_TIME global;

state Start{

Y\_WARM = OFF;

Y\_LAMP = OFF;

Y\_ALARM = OFF;

COOK\_TIME = 0;

start Start\_Button\_Controle;

start Cooking;

start Lights\_alarm;

start Clocks;

start Display\_work;

start ChangeTimeButtonControle;

start HourButtonControle;

stop;

}

}

proc Start\_Button\_Controle { /\* при нажатии кнопки пуска увеличивает время готовки \*/

from proc INIT K\_BUTTON\_START, COOK\_TIME;

state PressControl{

if (K\_BUTTON\_START == ON) {

COOK\_TIME += ONE\_MIN;

set next;

}

}

state PauseOnButtonBounce{

timeout 1s {

set state PressControl;

}

}

}

proc Lights\_alarm { /\*зажигает и гасит лампочку, когда это требуется \*/

from proc INIT K\_DOOR, Y\_WARM, Y\_LAMP;

state TermControleForStart{

if ((K\_DOOR == OPEN) || (Y\_WARM == ON)){

Y\_LAMP = ON;

}

else {

Y\_LAMP = OFF;

}

loop;

}

}

proc Cooking{

from proc INIT K\_DOOR, Y\_ALARM, Y\_WARM, COOK\_TIME;

state Start{

if (K\_DOOR == OPEN){

COOK\_TIME = 0;

} else

{

if (COOK\_TIME != 0){

Y\_WARM = ON;

set next;

}

}

}

state Cooking{

if (K\_DOOR == OPEN){

Y\_WARM = OFF;

set state WaitForDoorClose;

}

timeout (COOK\_TIME) {

Y\_WARM = OFF;

set next;

}

}

state VoiceAlarm {

Y\_ALARM = ON;

COOK\_TIME = 0;

timeout 1s {

Y\_ALARM = OFF;

set state Start;

}

}

state WaitForDoorClose{

if (COOK\_TIME == 0)

set state Start;

else

if (K\_DOOR != OPEN){

Y\_WARM = ON;

set state Cooking;

}

else set state WaitForDoorClose;

timeout 180000 {

COOK\_TIME = 0;

set state Start;

}

}

}

proc Clocks {

long Time global;

state INIT{

Time = 0;

set state Clocks;

}

state Clocks{

set state AddMinute;

}

state AddMinute {

timeout 60 \* 1s {

Time += ONE\_MIN;

set state Clocks;

}

}

}

proc Display\_work {

from proc INIT COOK\_TIME;

from proc Clocks Time;

long TimeTable local;

state Start{

TimeTable = Time;

set state ShowTime;

}

state ShowTime {

if (COOK\_TIME == 0){

TimeTable = Time;

}

else {

TimeTable = COOK\_TIME;

}

loop;

}

}

proc ChangeTimeButtonControle { /\* при нажатии кнопки пуска увеличивает время на 1 мин \*/

from proc INIT K\_BUTTON\_CHANGE\_TIME;

from proc Clocks Time;

state PressControl{

if (K\_BUTTON\_CHANGE\_TIME == ON) {

Time += ONE\_MIN;

set next;

}

}

state PauseOnButtonBounce{

timeout 1s set state PressControl;

}

}

proc HourButtonControle {

from proc INIT K\_BUTTON\_HOUR;

from proc Clocks Time;

state PressControl{

if (K\_BUTTON\_HOUR == ON) {

Time += ONE\_MIN \* 60;

set next;

}

}

state PauseOnButtonBounce{

timeout 1s set state PressControl;

}

}

proc Test {

long TestVar1 global;

int TestVar2 global;

state testState1{

if (TestVar1 == 1) {

set next;

start Test2;

}

}

state testState2{

timeout 1s set state testState1;

}

}

proc Test2 {

from proc Test TestVar1;

state testState1{

if (TestVar1 == 2) {

set next;

stop Test;

}

}

state testState2{

timeout 100/\*ONE\_SEK\*/ set state testState1;

}

}

}

# ПРИЛОЖЕНИЕ Б тезисы МНСК

УДК 004.4’2

Разработка программного модуля визуализации диаграмм процессов по спецификации на языке Reflex.

С. Е. Беленькая

Новосибирский государственный университет

Институт автоматики и электрометрии СО РАН, г. Новосибирск

При разработке и поддержке программного обеспечения в рамках итеративной модели встает проблема реверсивного инжиниринга и рефакторинга кода. Возникает необходимость создания документации, в частности, необходимость построения диаграмм по коду. Для распространенных объектно-ориентированных языков (Java, C++, С# и др.) уже реализовано множество программных средств, автоматизирующих этот процесс.

В то же время, при промышленной автоматизации, написании управляющих алгоритмов и программировании встраиваемых систем используются специализированные средства и языки. Процесс-ориентированное программирование (ПОП) выигрывает по многим критериям, однако при работе с ним документация создается вручную, что, в случае построения диаграмм, занимает значительное время и может быть причиной ошибок. В связи с этим возникает потребность в автоматизации процесса создания и визуализации диаграмм для ПОП (в частности, для языка Reflex), поэтому целью работы является разработка программного модуля визуализации диаграмм процессов по спецификации на языке Reflex.

В ходе работы была выявлена специфика языка Reflex: программа состоит из описания процессов, взаимодействующих по данным и по управлению, которые представлены автоматами состояний, используется кооперативная модель многопоточности. На основе выявленных особенностей языка были разработаны графические нотации для отображения связей процессов по данным и управлению, основанные на диаграммах UML, также будет использоваться нотация диаграммы состояний UML. Наиболее поддерживаемый редакторами и обладающий возможностью конвертации в широкий спектр форматов, GML выбран форматом хранения диаграмм. Реализацию модуля решено выполнить в виде плагина Eclipse для дальнейшей интеграции в среду разработки Reflex. Предстоит провести тестирование и доработку модуля.

Автоматическая визуализация диаграмм позволяет ускорить рецензирование и рафакторинг кода, а также упрощает поддержку проектов.

Работа выполняется при поддержке JetBrains Research.

# ПРИЛОЖЕНИЕ В Программный модуль генерации диаграмм процессов по спецификации на языке Reflex Руководство оператора

Листов 6

Новосибирск 2020

## 

Оглавление

[АННОТАЦИЯ 53](#_Toc42637331)

[1 Назначение программного модуля 54](#_Toc42637332)

[1.1. Функциональное назначение программного модуля 54](#_Toc42637333)

[1.2. Эксплуатационное назначение программного модуля 54](#_Toc42637334)

[1.3. Функции модуля 54](#_Toc42637335)

[2 Условия выполнения программы 54](#_Toc42637336)

[2.1. Минимальный состав аппаратных средств 54](#_Toc42637337)

[2.2. Минимальный состав программных средств 55](#_Toc42637338)

[2.3. Требования к оператору 55](#_Toc42637339)

[3 Выполнение программы 55](#_Toc42637340)

[3.1. Загрузка и запуск программного модуля 55](#_Toc42637341)

[3.2. Выполнение программного модуля 55](#_Toc42637342)

[3.3. Завершение работы программного модуля 55](#_Toc42637343)

[4 Сообщения оператору 55](#_Toc42637344)

[5 Лист регистрации изменений 56](#_Toc42637345)

## АННОТАЦИЯ

В данном программном документе приведено руководство оператора по применению и эксплуатации программного модуля визуализации диаграмм процессов по спецификации на языке Reflex. В данном документе, в разделе «Назначение программы» указаны сведения о назначении программного модуля и информация, необходимая для понимания функциональности программного модуля и принципов ее эксплуатации. В разделе «Условия выполнения программы» перечислены условия, являющиеся необходимыми для выполнения программного модуля. Раздел «Выполнение программы» содержит последовательность действий оператора, которые необходимы для загрузки, запуска, выполнения и завершения программного модуля, приведено описание функций, с помощью которых оператор осуществляет загрузку и управляет выполнением программного модуля, а также ответы программного модуля. В разделе «Сообщения оператору» приведены тексты сообщений, выдаваемых в ходе выполнения программного модуля, описание их содержания и соответствующие действия оператора (действия оператора в случае сбоя, возможности повторного запуска программы и т.п.).

Оформление программного документа «Руководство оператора» произведено по требованиям ГОСТ 19.505-79 «ЕСПД. Руководство оператора» и ГОСТ 19.105-78 «Единая система программной документации (ЕСПД). Общие требования к программным документам (с Изменением N 1)».

## Назначение программного модуля

### Функциональное назначение программного модуля

Программный модуль предназначен для автоматического построения диаграмм процессов (диаграммы состояний и связи процессов по данным и управлению) по спецификации на языке Reflex. Модуль строит диаграммы на основе абстрактного синтаксического дерева, которое создается транслятором языка Reflex и автоматически подается на вход модулю.

### Эксплуатационное назначение программного модуля

Программный модуль должен эксплуатироваться разработчиками управляющих алгоритмов на языке Reflex для автоматической генерации графической документации к исходному коду. Конечными пользователями должны являться разработчики на языке Reflex, аналитики и технические писатели.

### Функции модуля

Модуль реализует следующие функции:

- Построение диаграммы состояний каждого из описанных в исходном коде процессов в формате GML..

- Построение диаграммы связи процессов по данным в формате GraphML.

- Построение диаграммы связи процессов по управлению в формате GraphML.

- Автоматическое разделение не имеющих общих вершин диаграмм связи процессов по данным на разные файлы.

- Сохранение в вершинах процессов в диаграммах связи процессов по данным и по управлению ссылок на диаграмму состояний этого процесса.

# Условия выполнения программы

### 2.1. Минимальный состав аппаратных средств

Для работы программного модуля генерации диаграмм требуется персональный компьютер, включающий в себя:

Процессор, совместимый с архитектурой x86 с тактовой частотой 2 ГГц или выше;

Оперативную память объемом 4 Гб или выше;

Жесткий диск объемом 128 Гб или выше;

Графический адаптер: интегрированный в процессор графический процессор с тактовой частотой 1 ГГц или выше; допускается применение выделенного графического адаптера с тактовой частотой 1 ГГц или выше;

Монитор с разрешением экрана не менее 1024 на 768 пикселей;

Мышь.

### 2.2. Минимальный состав программных средств

Модуль генерации диаграмм интегрирован в среду разработки Reflex. Для его работы необходима установленная среда разработки Reflex.

### 2.3. Требования к оператору

Конечный оператор (пользователь) программного модуля должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы, а также практическими навыками работы с интерфейсом среды разработки на языке Reflex. Пользователь должен иметь знания в области процесс-ориентированного программирования на языке Reflex.

# Выполнение программы

### Загрузка и запуск программного модуля

Программный модуль автоматически запускается с помощью графического интерфейса среды разработки для языка Reflex (Reflex IDE): если в запущенной среде разработки, в которую интегрирован модуль, нажать кнопку сохранения файла исходного кода на языке Reflex, программный модуль будет запущен и начнет генерацию диаграмм процессов.

### Выполнение программного модуля

Программный модуль работает в фоновом режиме. После запуска он генерирует диаграммы состояний процессов, диаграммы связи процессов по данным и по управлению, записывая их в файлы формата GML и GraphML. Дополнительных действий оператора не требуется.

### Завершение работы программного модуля

Программный модуль автоматически закончит свою работу после того, как будут сгенерированы все файлы диаграмм процессов, которые будут находиться по умолчанию в папке проекта на языке Reflex, в подпапке src-gen.

# Сообщения оператору

В ходе работы программного модуля оператор может увидеть следующие сообщения в консоли Reflex IDE:

* Generate statechart GML diagrams...

Данное сообщение отображается, когда модуль начал этап генерации диаграмм состояний процессов. Дополнительных действий оператора не требуется.

* Done

Данное сообщение отображается, когда этап был успешно завершен (например, когда все диаграммы состояний процессов были успешно сгенерированы). Дополнительных действий оператора не требуется.

* Generate GraphML activity diagram...

Данное сообщение отображается, когда программный модуль начал этап генерации диаграммы связи процессов по управлению. Дополнительных действий оператора не требуется.

* Generate GraphML data diagram...

Данное сообщение отображается, когда программный модуль начал этап генерации диаграммы связи процессов по данным. Дополнительных действий оператора не требуется

# Лист регистрации изменений

Таблица 8 - Лист регистрации изменений в программном документе «Руководство оператора»

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа | Подпись | Дата |
| Номер изм. | измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

# ПРИЛОЖЕНИЕ Г Программный модуль визуализации диаграмм процессов по спецификации на языке Reflex Описание программы

Листов 12

Новосибирск 2020

## Cодержание

[АННОТАЦИЯ 59](#_Toc42636371)

[1 Общие сведения 60](#_Toc42636372)

[1.1 Обозначение и наименование программного модуля 60](#_Toc42636373)

[1.2 Программное обеспечение, необходимое для функционирования программного модуля 60](#_Toc42636374)

[1.3 Языки программирования 60](#_Toc42636375)

[2 Функциональное назначение 61](#_Toc42636376)

[2.1 Назначение программного модуля 61](#_Toc42636377)

[3 Описание логической структуры 62](#_Toc42636378)

[3.1 Структура программного модуля 62](#_Toc42636379)

[3.2 Алгоритм программного модуля 67](#_Toc42636380)

[3.3 Связи между составными частями программы 67](#_Toc42636381)

[3.4 Связи программы с другими программами 67](#_Toc42636382)

[4 Используемые технические средства 67](#_Toc42636383)

[5 Вызов и загрузка 68](#_Toc42636384)

[6 Входные данные 68](#_Toc42636385)

[7 Выходные данные 68](#_Toc42636386)

[8 Лист регистрации изменений 68](#_Toc42636387)

## АННОТАЦИЯ

В данном программном документе приведено описание программного модуля генерации диаграмм процессов по спецификации на языке Reflex, предназначенного для автоматической генерации диаграмм процессов по исходному коду программы на языке Reflex. Исходным языком программного модуля является Xtend. Средство разработки - интегрированная среда разработки Eclipse.

Основными функциями программного модуля является генерация файлов диаграмм состояний процессов и диаграмм связи процессов по данным и по управлению. В результате работы программного модуля создается набор файлов в форматах GML для диаграмм состояний процессов и в формате GraphML для диаграмм связи процессов по данным и по управлению, которые можно открыть одним из доступных программных средств визуализации графов.

Оформление программного документа «Описание прогрммы» произведено по требованиям ГОСТ 19.402-78 «ЕСПД. Описание программы» и ГОСТ 19.105-78 «Единая система программной документации (ЕСПД). Общие требования к программным документам (с Изменением N 1)».

## Общие сведения

### Обозначение и наименование программного модуля

Программный модуль визуализации диаграмм процессов по спецификации на языке Reflex имеет следующие атрибуты:

- Наименование исполняемого файла: ReflexGenerator.class;

- Размер исполняемого файла: 5.27 Кбайт;

- Версия файла: 1.0;

- Исходное имя файла: ReflexGenerator.class;

- Краткое описание программного модуля: модуль автоматического создания графической документации;

- Язык: английский.

### Программное обеспечение, необходимое для функционирования программного модуля

- среда разработки для языка Relex;

- операционная система, поддерживающая подходящую версию среды разработки Reflex.

### Языки программирования

Исходным языком программного модуля визуализации диаграмм процессов является Xtend. Средство разработки - интегрированная среда разработки Eclipse.

## Функциональное назначение

### Назначение программного модуля

Программный модуль предназначен для генерации диаграмм связи процессов по данным, по управлению и диаграмм состояний процессов на основании синтаксического дерева.

Программный модуль решает следующие задачи:

- Построение диаграммы состояний каждого из описанных в исходном коде процессов в формате GML.

- Построение диаграммы связи процессов по данным в формате GraphML.

- Построение диаграммы связи процессов по управлению в формате GraphML.

- Автоматическое разделение не имеющих общих вершин диаграмм связи процессов по данным на разные файлы.

- Сохранение в вершинах процессов в диаграммах связи процессов по данным и по управлению ссылок на диаграмму состояний этого процесса.

## Описание логической структуры

Архитектура программного модуля была спроектирована на основе паттерна Model-View-Controller с активной моделью.

### Структура программного модуля

Программный модуль состоит из тринадцати классов:

* Класс ReflexGenerator выполняет роль контроллера и запускается по умолчанию.
* Класс ProcessDiagramGenerator – суперкласс для StatechartDiagramGenerator и ProcessDiagramGraphMLGenerator, определяет базовое поведение генератора модели диаграммы.
* Класс StatechartDiagramGenerator отвечает за создание модели диаграммы состояний процесса.
* Класс ActiveProcess определяет одно ребро диаграммы.
* Класс ProcessDiagramGraphMLGenerator – суперкласс для ActivityDiagramGenerator и DataDiagramGenerator. Задает базовое поведение генератора моделей разделяемых диаграмм.
* Класс DataDiagramGenerator отвечает за создание модели диаграммы связи процессов по данным.
* Класс ActivityDiagramGenerator отвечает за создание модели диаграммы связи процессов по управлению.
* Класс DiagramNode задает одну вершину диаграммы с ее характеристиками.
* Класс GMLTextGenerator отвечает за генерацию файла диаграммы в формате GML.
* Класс GraphMLTextGenerator отвечает за генерацию файла диаграммы в формате GraphML.
* - Класс GraphSeparator осуществляет разбиение модели диаграммы на несколько, если они не имеют общих вершин.
* Класс ProcessId описывает один процесс.
* Класс ResultOfSeparation определяет возвращаемый формат модуля разбиения моделей диаграмм.

Взаимодействие классов в рамках шаблона MVC представлено на рис. 5.

Далее в таблицах 9-16 перечислены входящие в наиболее информативные классы функции.

Таблица 9 – Описание методов класса ReflexGenerator

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Результат | Назначение |
| doGenerate | Resource resource, IFileSystemAccess2 fsa, IGeneratorContext context | void | Вызывается по умолчанию. Выполняет функции контроллера: вызывает генерацию моделей и модуль их разделения |

Таблица 10 – Описание методов класса StatechartDiagramGenerator

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Результат | Назначение |
| getStatechartNodes | Process process | void | Создает модель вершин диаграммы состояний процесса |
| generateStatechartDiagramModel | Resource resource, Process process | void | Создает модель диаграммы состояний процесса |
| generateStatechartDiagram | void | String | Вызывает генерацию файла диаграммы состояний процесса путем вызова GmlTextGenerator |
| generateStatechartModel | Resource resource, Process process | void | Создает модель ребер диаграммы состояний процесса |
| getStatechartList | Statement statement, int contextStateId, String contextLabel, String expressionStatement | ArrayList<ActiveProcess> | Полиморфный метод, возвращающий информацию о связи между состояниями |
| getContextLabel | ExpressionStatement statement, int contextStateId, String contextLabel, String expressionStatement | String | Полиморфный метод, возвращающий выражения, которые выполняются при переходе между состояниями |
| translateExpr | Expression expr | String | Возвращает текстовое представление выражения |

  Таблица 11 – Описание методов класса ProcessDiagramGraphMLGenerator

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Результат | Назначение |
| generateProcList | Resource resource | void | Создает модель вершин диаграммы |
| separateDiadram | void | ResultOfSeparation | Вызывает модуль разделения моделей |

  Таблица 12 – Описание методов класса DataDiagramGenerator

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Результат | Назначение |
| getVariablesNodes | Resource resource | void | Создает модель вершин переменных диаграммы связи процессов по данным |
| generateDataModel | Resource resource | void | Создает модель ребер диаграммы связи процессов по данным |
| getVariableName | DeclaredVariable/ImportedVariable variable | ArrayList<String> | Полиморфный метод. Возвращает имена переменных |
| getVariableAction | DeclaredVariable/ImportedVariable variable | String | Полиморфный метод. Возвращает тип взаимодействия с переменной |
| getVariableNameAndType | ProgramVariable/ PhysicalVariable variable | String | Полиморфный метод. Возвращает имя переменной и ее тип |
| getSigned | ReflexType/ CType type | String | Полиморфный метод. Возвращает модификатор типа |
| generateDataDiagramModel | Resource resource | void | Генерирует модель диаграммы |
| generateDataGraphMLDiagram | HashMap<String, DiagramNode> nodesId, ArrayList<ActiveProcess> diagramModel, String url, String statechartFileNameTail | String | Вызывает генерацию файла диаграммы связи процессов по данным путем вызова GraphMLTextGenerator |

Таблица 13 – Описание методов класса ActivityDiagramGenerator

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Результат | Назначение |
| generateActivityModel | Resource resource | void | Создает модель ребер диаграммы связи процессов по управлению |
| getActiveList | Statement statement, int contextProcessId | ArrayList<ActiveProcess> | Полиморфный метод. Определяет связи между вершинами диаграммы |
| generateActivityDiagramModel | Resource resource | void | Создает модель диаграммы связи процессов по управлению |
| generateActivityGraphMLDiagram | HashMap<String, DiagramNode> nodesId, ArrayList<ActiveProcess> diagramModel, String url, String statechartFileNameTail | String | Вызывает генерацию файла диаграммы связи процессов по управлению путем вызова GraphMLTextGenerator |

  Таблица 14 – Описание методов класса GMLTextGenerator

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Результат | Назначение |
| writeHeadGML | ProcessDiagramGenerator generator | String | Возвращает строку, содержащую заголовок выходного GML файла |
| generateNodeGraphics | int nameLength, String typeOfNode | String | Генерирует строку с настройками отображения вершины диаграммы |
| generateLabelGraphics | String label | String | Генерирует LabelGraphics, задающий настройки щрифта и текст подписи внутри вершины |
| generateOneProcessNode | int processId, String processName, String typeOfNode | String | Возвращает объявление одной вершины в GML графе |
| generateOneEdge | int fromId, int toId, String label | String | Возвращает описание ребра в формате GML с заданными в параметрах свойствами |
| generateAllEdges | ArrayList<ActiveProcess> procList | String | Возвращает описание всех ребер графа диаграммы процессов |
| generateNodes | (ProcessDiagramGenerator generator | String | Возвращает объявления всех вершин создаваемой диаграммы процессов |

  Таблица 15 – Описание методов класса GraphMLTextGenerator

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Результат | Назначение |
| headGraphMlGenerator | ProcessDiagramGenerator generator | String | Возвращает заголовок выходного GraphML файла |
| generateShapeNodeGraphML | int nameLength, String processName, String shapetype | String | Задает настройки отображения вершин диаграммы |
| dataKeysGraphMLGenerator | int nameLength, String processName, String shapetype, String urlStatechartDiagram | String | Добавляет необходимые заголовки тегов |
| nodeGraphMLGenerate | int processId, String processName, String shapetype, String urlStatechartDiagram | String | Возвращает объявление одной вершины в GraphML графе |
| generateOneEdgeGraphML | int edgeId, int fromId, int toId, String edgeLabel | String | Возвращает описание ребра в формате GraphML |
| generatePoliLineEdgeGraphML | String edgeLabel | String | Задает настройки отображения одного ребра |
| generateAllEdges | ArrayList<ActiveProcess> procList | String | Возвращает описание всех ребер графа диаграммы процессов |
| generateNodes | ProcessDiagramGenerator generator, String url, String statechartFileNameTail | String | Возвращает объявления всех вершин создаваемой диаграммы процессов |

  Таблица 16 – Описание методов класса GraphSeparator

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Параметры | Результат | Назначение |
| separateGraph | ArrayList<ActiveProcess> procList, HashMap<String, DiagramNode> nodeIdList | ResultOfSeparation | Разделяет модели диаграммы, если у них нет общих вершин |
| getElementList | ArrayList<ActiveProcess> procList, HashMap<String, DiagramNode> nodeIdList, DiagramNode element | ArrayList <DiagramNode> | Возвращает список потомков вершины |
| getActiveListByNode | ArrayList<ActiveProcess> procList, DiagramNode element | ArrayList<ActiveProcess> | Возвращает список ребер, исходящих из вершины |
| setVisitedForList | HashMap<String, DiagramNode> tempNodeList, int component | void | Задает номер компоненты для списка вершин |
| getElementNodeIdList | HashMap<String, DiagramNode> nodeIdList, int index | DiagramNode | Возвращает вершину диаграммы по ее индексу |

### Алгоритм программного модуля

Сначала контроллер проходит по списку процессов и запускает создание модели диаграммы состояний текущего процесса по абстрактному синтаксическому дереву программы, потом запускает генератор текста диаграммы в формате GML.

После этого контроллер запускает генерацию модели диаграмм связи процессов по данным и по управлению на основе абстрактного синтаксического дерева, полученного модулем на вход. Далее он передает полученные модели в модуль разделения диаграмм, который разбивает модель на независимые части (то есть, если в одной модели оказалось две диаграммы, не имеющие общих вершин, они будут разделены на две разные модели).

После получения списка разделенных на независимые части моделей контроллер запускает генерацию текста диаграмм в формате GraphML, при этом в каждую вершину процесса помещается ссылка на диаграмму состояний этого процесса, сгенерированную ранее.

Алгоритм работы модуля разделения диаграмм процессов, не имеющих общих вершин, описан ниже. Не посещенные вершины процессов добавляются в очередь. Далее достаем последовательно из очереди один элемент. После этого:

* Если он еще не был посещен, то отмечаем его как посещенный и добавляем в текущую компоненту. Получаем список всех его потомков и добавляем их в очередь.
* Если он уже был посещен.

Если он относится к компоненте, отличной от текущей компоненты, то:

* + Если номер компоненты меньше текущей.

Номер текущей компоненты заменяется на номер найденной компоненты, все вершины текущей компоненты добавляются к вершинам найденной.

* + Если номер компоненты больше текущей.

Номер текущей компоненты остается тем же, все элементы найденной компоненты добавляются к текущей компоненте.

### Связи между составными частями программы

Контроллер запускает генерацию моделей диаграмм. Классы, отвечающие за модель, вызывают модуль генерации текста диаграмм (View).

### Связи программы с другими программами

Программный модуль генерации диаграмм процессов связан со средой разработки на языке Reflex.

## Используемые технические средства

Программный модуль эксплуатируется на персональном компьютере (ПК) типа IBM. Режим работы – в формате приложения, работающего в фоновом режиме, без взаимодействия с оператором. Входные и выходные данные хранятся на жестком диске.

Требования, предъявляемые к ПК для корректной работы программного модуля:

* Процессор, совместимый с архитектурой x86 с тактовой частотой 2 ГГц или выше;
* Оперативная память объемом 4 Гб или выше;
* Жесткий диск объемом 128 Гб или выше;
* Графический адаптер: интегрированный в процессор графический процессор с тактовой частотой 1 ГГц или выше; допускается применение выделенного графического адаптера с тактовой частотой 1 ГГц или выше;
* Монитор с разрешением экрана не менее 1024 на 768 пикселей;
* Мышь.

## Вызов и загрузка

Программный модуль автоматически запускается с помощью графического интерфейса среды разработки для языка Reflex (Reflex IDE): если в запущенной среде разработки, в которую интегрирован модуль, нажать кнопку сохранения файла исходного кода на языке Reflex, программный модуль будет запущен и начнет генерацию диаграмм процессов.

Точкой входа является метод doGenerate(Resource resource, IFileSystemAccess2 fsa, IGeneratorContext context) класса ReflexGenerator.

## Входные данные

В качестве входных данных используется синтаксическое дерево, созданное транслятором языка Reflex.

## Выходные данные

В качестве выходных данных создаются файлы диаграмм состояний процессов в формате GML и файлы диаграмм связи процессов по данным и по управлению в формате GraphML.

## Лист регистрации изменений

Таблица 17 – Лист регистрации изменений в программном документе «Описание программы»

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Лист регистрации изменений | | | | | | | | | |
| Номера листов (страниц) | | | | | Всего листов (страниц) в документе | № документа | Входящий № сопроводительного документа | Подпись | Дата |
| Номер изм. | измененных | замененных | новых | аннулированных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |