



Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехники и комплексной автоматизации»

КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине «Модели и методы анализа проектных решений»

на тему

«Проектирование структуры графовых моделей в программном комплексе
GBSE»

Студент РК6-71Б
группа

Руководитель КП


подпись, дата

Трипин И.В.
ФИО

подпись, дата

Соколов А.П.
ФИО

Москва, 2021

Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ

Заведующий кафедрой РК-6
индекс

А.П. Карпенко

«___» _____ 2021 г.

ЗАДАНИЕ

на выполнение курсового проекта

Студент группы: РК6-71Б

Тришин Илья Вадимович

(фамилия, имя, отчество)

Тема курсового проекта : Проектирование структуры графовых моделей в программном комплексе GBSE

Источник тематики (кафедра, предприятие, НИР): кафедра

Тема курсового проекта утверждена на заседании кафедры «Системы автоматизированного проектирования (РК-6)», Протокол № _____ от «___» _____ 2021 г.

Техническое задание

Часть 1. Аналитический обзор литературы.

В рамках аналитического обзора литературы должно быть приведено описание некоторого программного комплекса или разработки, в которой вычислительные процессы в пределах решения одной задачи также организованы в виде графа. Кроме того, целесообразно провести сравнение данной разработки с GBSE.

Часть 2. Постановка задачи

Должен быть изучен текущий синтаксис языка aDot и та часть исходного кода библиотеки comsdk, которая отвечает за формирование графовой модели по её описанию на языке aDot и последующий её обход. На основании этого должен быть сформирован список требований к новой версии этой части библиотеки.

Часть 3. Программная архитектура

Должна быть разработана программная архитектура модуля формирования графовых моделей в comsdk и предоставлено её описание в виде диаграмм на языке UML

Оформление курсового проекта :

Расчетно-пояснительная записка на 26 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.):

<i>количество: 7 рис., 0 табл., 13 источн.</i>
<i>[здесь следует ввести количество чертежей, плакатов]</i>

Дата выдачи задания «07» октябрь 2021 г.

Студент


подпись, дата

Тришин И.В.
ФИО

Руководитель курсового проекта

подпись, дата

Соколов А.П.
ФИО

Примечание: Задание оформляется в двух экземплярах: один выдается студенту, второй хранится на кафедре.

РЕФЕРАТ

курсовой проект : 26 с., 9 глав, 7 рис., 0 табл., 13 источн.

.

Данная работа направлена на разработку новой программной архитектуры фрагмента ядра программного комплекса GBSE, связанного с внутренним представлением ориентированного графа, в виде которого представляется процесс решения некоторой сложной вычислительной задачи.

Тип работы: курсовой проект .

Тема работы: *«Проектирование структуры графовых моделей в программном комплексе GBSE».*

Объект исследования: графоориентированный программный каркас.

Основная задача, на решение которой направлена работа: @Основная задача, на решение которой направлена работа@.

Цели работы состоят в: выявить существующие недостатки в реализации описанного метода и предложить программную архитектуру, которая бы позволила их устранить

В результате выполнения работы: 1) описаны основные положения графоориентированного подхода; 2) проведено сравнение программных комплексов, в которых вычислительные процессы для решения определённой задачи организуются в виде графа; 3) сформированы требования к представлению графовых моделей и их обходу в пределах программного комплекса GBSE; 4) изучена и проверена на соответствие требованиям программная архитектура программного каркаса GBSE; 5) предложена новая программная архитектура, которая отвечает сформированным требованиям.

СОКРАЩЕНИЯ

aDOT Расширенный формат DOT (описание представлено в [1]). 13, 16

aINI Расширенный формат INI (описание представлено в [2]). 14

CASE Computer-aided Software Engineering. 7

DFD диаграмма потоков данных (Data Flow Diagram). 7

GBSE Graph-based Software Engineering. 7

ТО технический объект, в т.ч. сложный процесс, система. 9, 13

СОДЕРЖАНИЕ

СОКРАЩЕНИЯ	5
ВВЕДЕНИЕ	7
1. Сравнительная характеристика программных комплексов, реализующих графоориентированный подход	9
1.1. Выделение признаков для сравнения	9
1.2. Описание сравниваемых комплексов	10
1.3. Сравнительная таблица	12
2. Постановка задачи	16
2.1. Требования к графовому модулю comsdk	16
2.2. Анализ текущей версии графового модуля comsdk	18
3. Архитектура программной реализации	20
3.1. Узлы и рёбра графа	20
3.2. Хранение узлов и рёбер графа	21
ЗАКЛЮЧЕНИЕ	24
Литература	25

ВВЕДЕНИЕ

При проведении современных исследований возникает необходимость автоматизировать процессы решения сложных вычислительных задач. Достижение подобной цели не представляется возможным без формально определенного метода организации процессов в автоматизированной системе. Проектирование, создание и сопровождение подобных систем является трудоемкой задачей, для решения которой применяют инструментальные средства и среды разработки автоматизированных систем (**CASE**-системы)[3] В некоторых очень узко направленных системах подобная организация напрямую зависит от поставленной задачи. В более универсальных же системах, о которых пойдёт речь в этой работе, разрабатывается особая архитектура, которая позволяет организовать различные процессы для решения различных задач по-разному. Подходы к построению данной архитектуры освещены, помимо прочего, в [4].

В данной работе рассматривается, в первую очередь т.н. графоориентированный подход к решению задач проектирования. Данный подход подразумевает организацию различных вычислительных процессов в виде графа. Многие известные методики предполагают организацию вычислительных процессов в графовой форме. Например, применяют: диаграммы потоков данных (**DFD**), граф-схемы, конечные автоматы, диаграммы перехода состояний. Такое описание позволяет выделять структурные единицы приложения в виде функций или подпрограмм и связывать их между собой в определенной последовательности[5].

Одной из реализаций данного подхода является разработка Соколова А.П. и Першина А.Ю. графоориентированный программный каркас для решения задач проектирования **GBSE**. Данный программный каркас направлен на упрощение и структуризацию разработки прикладного программного обеспечения для решения описанных выше задач.

Определение 1. *Графовой моделью* процесса решения задачи назовём некоторое формализованное описание этого решения, где отдельные вычислительные процессы организуются в виде графа.

При описании графовых моделей в **GBSE** вводятся следующие понятия:

- *состояние данных* – некоторый строго определённый набор именованных переменных фиксированного типа, характерных для решаемой задачи;
- *морфизм* – некоторое отображение одного состояния данных в другое;
- *функция-предикат* – функция, определяющая соответствие подаваемого ей на вход набора данных тому виду, который требуется для выполнения отображения;
- *функция-обработчик* – функция, отвечающая за преобразование данных из одного состояния в другое;
- *функция-селектор* – функция, отвечающая в процессе обхода графовой модели за выбор тех рёбер, которые необходимо выполнить на следующем шаге в соответствии с некоторым условием.

Более подробное теоретическое описание концепции, реализованной в GBSE представлено в [6]. В данной работе внимание сосредоточено в первую очередь на программной реализации этой концепции в библиотеке comsdk для языка программирования C++.

1 Сравнительная характеристика программных комплексов, реализующих графоориентированный подход

Была проведена сравнительная характеристика рассматриваемой разработки с представленными в настоящее время на рынке продуктами. В рамках сравнения были рассмотрены известные программные комплексы, в основе которых в том или ином виде лежит идея организации вычислений, описываемая с помощью ориентированных графов.

Для проведения сравнения с программным каркасом GBSE выбирались программные продукты, в которых так или иначе реализован описанный выше подход. В первую очередь был рассмотрен программный комплекс pSeven, разработанный отечественной компанией DATADVANCE. Он направлен в первую очередь на решение конструкторских, оптимизационных задач и, помимо этого, задач анализа данных, что в первом приближении делает его аналогом GBSE по предметному назначению. У автора работы присутствует опыт работы с этим комплексом в рамках прохождения курса лабораторных работ по изученной на кафедре дисциплине "Методы оптимизации". В данном курсе были освещены основы работы с pSeven и основные принципы организации вычислений в нём. Таким образом, на момент выбора программных комплексов для сравнения уже имелась некоторая информация о pSeven, которая позволила включить его в рассмотрение.

Кроме того, научным руководителем данной работы была рекомендована разработка отечественной компании "Ладуга" PRADIS - комплекс, так же направленный на решение конструкторских задач. В данной разработке основной упор сделан на задачи анализа проектных решений на микро- и макроуровне.

1.1 Выделение признаков для сравнения

Среди прочих должны были быть выделены признаки, относящиеся как к общей структуре программного комплекса, так и к особенностям реализации в нём графоориентированного подхода и, кроме того, к особенностям взаимодействия с пользователем при решении задач, требующих действий с его стороны.

Сравнение осуществлялось с учётом следующих характерных признаков:

1. предметное назначение;
2. принципы формирования графовых моделей;
3. формат описания графовых моделей;
4. файловая структура проекта проведения анализа **ТО**;
5. особенности работы с входными и выходными данными графовых моделей;
6. особенности передачи данных между узлами графовых моделей;
7. поддержка ветвлений и циклов в топологии графовых моделей;
8. поддержка параллельной обработки данных;

9. возможность выбрать из набора однотипных промежуточных результатов расчётов некоторые экземпляры и продолжить расчёт только для них;
10. возможность доопределять входные данные непосредственно во время обхода графовой модели.

1.2 Описание сравниваемых комплексов

1.2.1 Программный комплекс Pradis

Программный комплекс Pradis, разработанный отечественной компанией «Ладуга», предназначен для анализа динамических процессов в системах разной физической природы (механических, гидравлических и т.д.). Как правило, при помощи данного комплекса решаются нестационарные нелинейные задачи, в которых характеристики системы зависят от времени и пространственных координат. Круг задач, которые могут быть решены с помощью Pradis, достаточно широк: возможен анализ любых технических объектов, модели поведения которых представимы системами обыкновенных дифференциальных уравнений (СОДУ). Анализ статических задач обеспечивается как частный случай динамического расчета.

Практические возможности по решению конкретных задач определяются текущим составом библиотек комплекса, прежде всего библиотек моделей элементов [7]. Данный комплекс был рекомендован к обзору и сравнению, однако, после проведённого обзора официальной документации [8] не было получено достаточного представления об использовании элементов графоориентированных подходов в данном комплексе, поэтому было принято решение исключить его из дальнейшего рассмотрения.

1.2.2 Программный комплекс pSeven

В программном комплексе pSeven, разработанном компанией DATADVANCE, используется методология диаграмм потоков данных (англ. Dataflow diagram, DFD). В комплексе применяются ориентированные графы (орграфы) для описания процесса решения некоторой задачи проектирования технического объекта. В терминах pSeven: графовое описание процесса решения задачи называется *расчетной схемой* (англ. workflow), что, помимо прочего, соответствует классическому термину из области математического моделирования; узлам орграфа поставлены в соответствие процессы обработки данных (используется термин *блоки*), а рёбра определяют *связи* между блоками и направления передачи данных между процессами [9].

Используются следующие базовые понятия pSeven:

- **расчётная схема** – формальное описание процесса решения некоторой задачи в виде орграфа;
- **блок** – программный контейнер для некоторого процесса обработки данных, входные и выходные данные для которого задаются через порты (см. ниже);

- **порт** – переменная конкретного¹ типа, определённая в блоке и имеющая уникальное имя в его пределах;
- **связь** – направленное соединение типа “один к одному” между выходным и входным портами разных блоков.

С учётом данных понятий можно описать используемую методологию диаграмм потоков данных следующим образом. Расчётная схема содержит в себе набор процессов обработки данных (блоков), каждый из которых имеет (возможно, пустой) набор именованных входов и выходов (портов). Данные передаются через связи. Для избежания т.н. гонок данных (англ. data races) множественные связи с одним и тем же входным портом не поддерживаются. Для начала выполнения каждому блоку требуются данные на всех входных портах. Все данные на выходных портах формируются по завершении исполнения блока [9].

Описанный подход наглядно иллюстрирует рисунок 1.

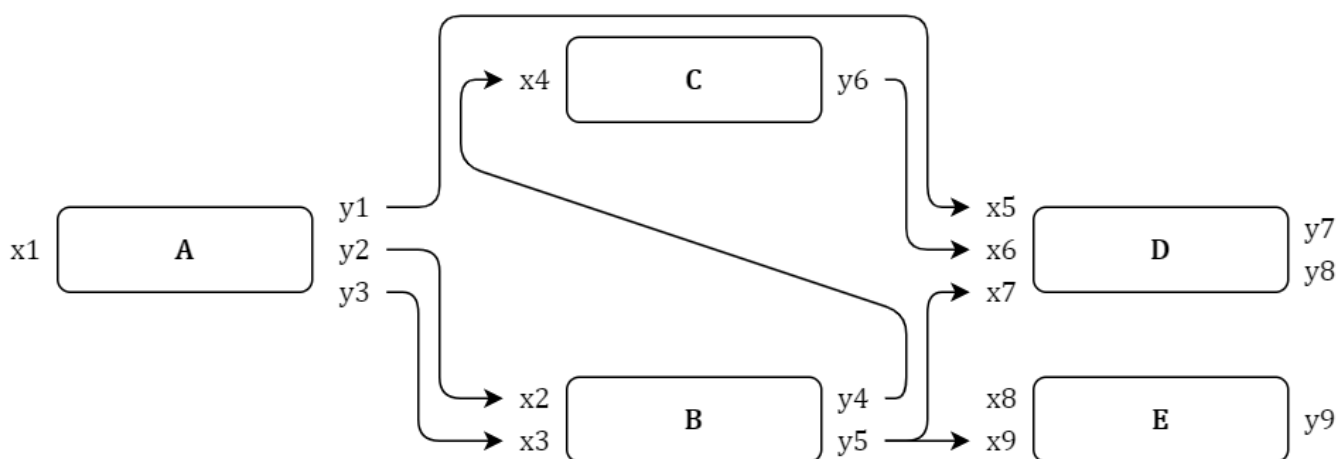


Рисунок 1. Пример диаграммы потоков данных

На рисунке 1 A, B, C, D - блоки обработки информации, x_i - входные порты, y_i - выходные. Стрелками показаны связи. Согласно изложенному выше принципу, сначала будет запущен блок A , по его завершении - блок B . Затем - блоки C и E (параллельно). По завершении блока C будет запущен блок D . На этом обход расчётной схемы завершится, и значения y_7, y_8 и y_9 будут сохранены в локальной базе данных.

Замечание 1 (принцип обхода в pSeven). Все порты, которые не привязаны к другим блокам, автоматически становятся внешними входами и выходами для всей расчётной схемы. Для начала обхода расчётной схемы должен быть предоставлен набор входных данных и указаны внешние выходные порты, значения которых обязательно должны быть вычислены в результате обхода. Обход производится в несколько этапов: сперва отслеживаются пути от необязательных выходных портов к входным, все встреченные на пути блоки помечаются, как неактуальные и не будут выполнены в дальнейшем; затем отслеживаются пути от обязательных выходных портов к входным и все встреченные на пути блоки помечаются, как обязательные к исполнению.

¹Динамическая типизация не поддерживается.

Наконец, обязательные к исполнению блоки запускаются, начиная с тех, которые подключены к внешним входам расчётной схемы, а неактуальные игнорируются. Обход прекращается, когда не остаётся необходимых для выполнения блоков [9].

По завершении обхода расчётной схемы результаты расчётов сохраняются в локальной базе данных проекта. В pSeven встроен инструмент для визуализации и анализа результатов. Схемы, графики, гистограммы и результаты анализа сохраняются в т.н. отчётах - специальных файлах, сохраняемых в директории проекта.

1.3 Сравнительная таблица

Результаты проведённого сравнения представлены в таблице 1.

Таблица 1. Сравнительная таблица

№	Признак	pSeven	GBSE
1	Предметное назначение	Задачи оптимизации, анализ данных	Задачи автоматизированного проектирования, алгоритмизация сложных вычислительных методов, анализ данных
2	Принцип формирования графовых моделей	Узлы – блоки (процессы), рёбра – связи (направление передачи данных) [9].	Узлы – состояния данных, рёбра – переходы между состояниями, с указанием функций перехода [6].
3	Формат описания орграфа	Расчетная схема (в форме орграфа) сохраняется в двоичный файл закрытого формата с расширением .p7wf.	Графовая модель (определяет алгоритм проведения комплексных вычислений в форме орграфа) сохраняется в текстовом файле открытого формата, подготовленного на языке aDOT[1], являющегося “сужением” (частным случаем) известного формата DOT (Graphviz).
4	Файловая структура проекта проведения анализа ТО	Проект состоит из непосредственно файла проекта, в котором хранятся ссылки на созданные расчётные схемы и локальную базу данных, сами расчётные схемы, файлы с их входными данными, файлы отчётов, где сохраняются выходные данные последних расчётов и результаты их анализа.	Проект состоит из .aDOT файла с описанием графа, .aINI-файлов с описанием форматов входных данных, библиотек функций-обработчиков, функций-предикатов и функций-селекторов, файлов, куда записываются выходные данные.

5	особенности работы с входными и выходными данными графовых моделей	Входные данные должны быть указаны при настройках внешних входных портов расчётной схемы. Данные с выходных портов схемы сохраняются в локальной базе данных. Для их записи в файлы для обработки/анализа вне pSeven необходимо воспользоваться специально предназначенными для этого блоками.	Входные данные хранятся в файле в формате aINI [2], откуда считываются при запуске обхода графа [10]. Для записи выходных/промежуточных данных в файлы или базы данных необходимо добавить соответствующие функции-обработчики. Формат выходных данных не регламентирован.
6	Особенности передачи параметров между узлами графовых моделей	Данные между узлами передаются согласно определённым связям, которые на уровне выполнения создают пространство в памяти для ввода и вывода данных для выполняемых в раздельных процессах блоков. Транзитная передача данных, которые не изменяются в данном блоке, на выход невозможна.	Поскольку узлами графа являются состояния данных, существует возможность задействовать в расчётах только часть данных, оставляя их другую часть неизменной.
7	Поддержка ветвлений и циклов	Присутствует. Достигается за счёт специальных управляющих блоков, которые отслеживают выполнение условий: для ветвления используется блок "Условие"(англ. condition), который перенаправляет данные на один из выходных портов в зависимости от выполнения описанного условия (подробнее см. [11]); Для реализации циклов в общем случае используются блоки "Цикл"(англ. loop)[12], но для некоторых задач существуют специализированные блоки, организующие логику работы цикла (например, блок "Оптимизатор"(англ. optimizer))	Присутствует по умолчанию

8	Поддержка параллельной обработки данных	Присутствует. Блоки, входящие в состав различных ветвлений схемы могут быть выполнены параллельно, поскольку они не зависят друг от друга по используемым данным.	Присутствует. Существует возможность обойти различные ветвления графа одновременно.
9	Возможность выбрать из набора однотипных промежуточных результатов расчётов некоторые экземпляры и продолжить расчёт только для них;	Производится на этапе анализа результатов с помощью отчётов, где можно задать фильтрацию выходных данных согласно указанным критерия. В случае, если результаты являются промежуточными, расчётную схему приходится разбивать на части.	Планируется реализовать средство визуализации данных, которое в совокупности с автоматической генерацией форм ввода[10] позволят отбирать корректные результаты промежуточных вычислений во время обхода графовой модели.
10	Возможность доопределения значений входных данных в процессе обхода графа	Отсутствует	Частично реализована при помощи функций-обработчиков специального типа, создающих формы ввода

2 Постановка задачи

2.1 Требования к графовому модулю comsdk

Концептуальная постановка задачи.

Для описания процесса решения задачи в виде графа в программном каркасе GBSE был разработан специальный текстовый формат **aDOT**. Он построен на основе распространённого языка описания графов Dot (Graphviz)[13]. Основным его отличием является необходимость хранить помимо узлов, рёбер и их атрибутов дополнительную информацию о вызываемых в процессе обхода графовой модели функциях, а именно:

- тип функции – обработчик, предикат, селектор;
- путь к библиотеке, из которой можно извлечь данную функцию;
- идентификатор.

Подробное описание этого формата приведено в [1]. Ниже приведён пример описания графовой модели на языке aDot (листинг 2.1)

Листинг 2.1. Пример описания графовой модели на языке aDot

```
1 digraph SIMPLEST {
2   FUNCA [module=libtest, entry_func=IncA]
3   FUNCB [module=libtest, entry_func=IncB]
4
5   CHECKA [module=libtest, entry_func=CheckAEq4]
6   CHECKB [module=libtest, entry_func=CheckBEq4]
7
8   SETA [module=libtest, entry_func=SetAEq1]
9   SETB [module=libtest, entry_func=SetBEq1]
10
11  PASS [module=libtest, entry_func=PassFunc]
12  PRED [module=libtest, entry_func=PassPred]
13
14  INCR_A [predicate=PRED, function=FUNCA]
15  INCR_B [predicate=PRED, function=FUNCB]
16  CH_A [predicate=CHECKA, function = PASS]
17  SET_A [predicate=PRED, function=SETA]
18  SET_B [predicate=PRED, function=SETB]
19  CH_B [predicate=CHECKB, function = PASS]
20
21  __BEGIN__ -> ROT [morphism=SET_A]
22  ROT -> ROOT[morphism=SET_B]
23  ROOT -> BR1, BR2 [morphism=(INCR_A, INCR_B)]
24  BR1 -> BR1_ST [morphism=INCR_A]
```



```

25  BR2 -> BR2_ST [morphism=INCR_B]
26  BR1_ST, BR2_ST -> MERGE [morphism=(INCR_A, INCR_B)]
27  MERGE -> __END__, __END__ [morphism=(CH_A, CH_B)]
28  }

```

В данном описании объявляются функции-обработчики PASS, FUNCA, FUNCB, SETA и SETB и функции-предикаты CHECKA, CHECKB и PRED, которые можно найти в библиотеке libtest. Кроме того, объявляются морфизмы, содержащие предикаты и обработчики. На рисунке 2 изображено визуальное представление модели, описанной выше с некоторыми пояснениями.

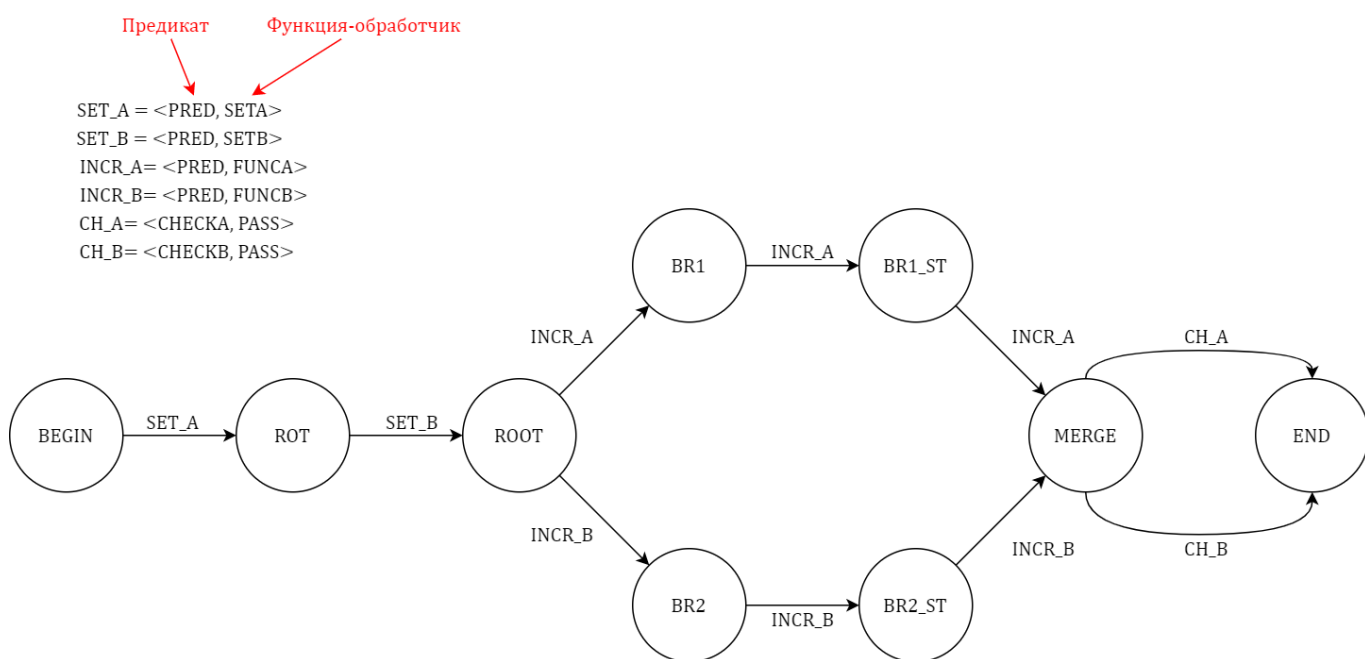


Рисунок 2. Пример графовой модели

В результате анализа текущего синтаксиса языка aDOT были сформулированы следующие требования к структуре графовых моделей в новой версии графового модуля библиотеки comsdk:

- 1) Каждое ребро графа должно иметь возможность привязать к нему до трёх морфизмов – препроцессор, обработчик и постпроцессор;
- 2) Каждый морфизм должен содержать в себе функцию-предикат и функцию-обработчик;
- 3) Каждый узел графа должен хранить состояние данных (т.е. сведения о типах и именах переменных);
- 4) Каждый узел графа должен иметь возможность привязать к нему функцию-селектор;
- 5) Каждый узел графа должен хранить данные о стратегии выполнения рёбер, исходящих из него (поочерёдное выполнение, выполнение в отдельных потоках, выполнение в отдельных процессах, выполнение на удалённом узле через SSH-подключение);

2.2 Анализ текущей версии графового модуля comsdk

На рисунке 3 представлена UML-диаграмма классов, связанных с представлением в comsdk ориентированного графа, описывающего организацию вычислительных процессов.

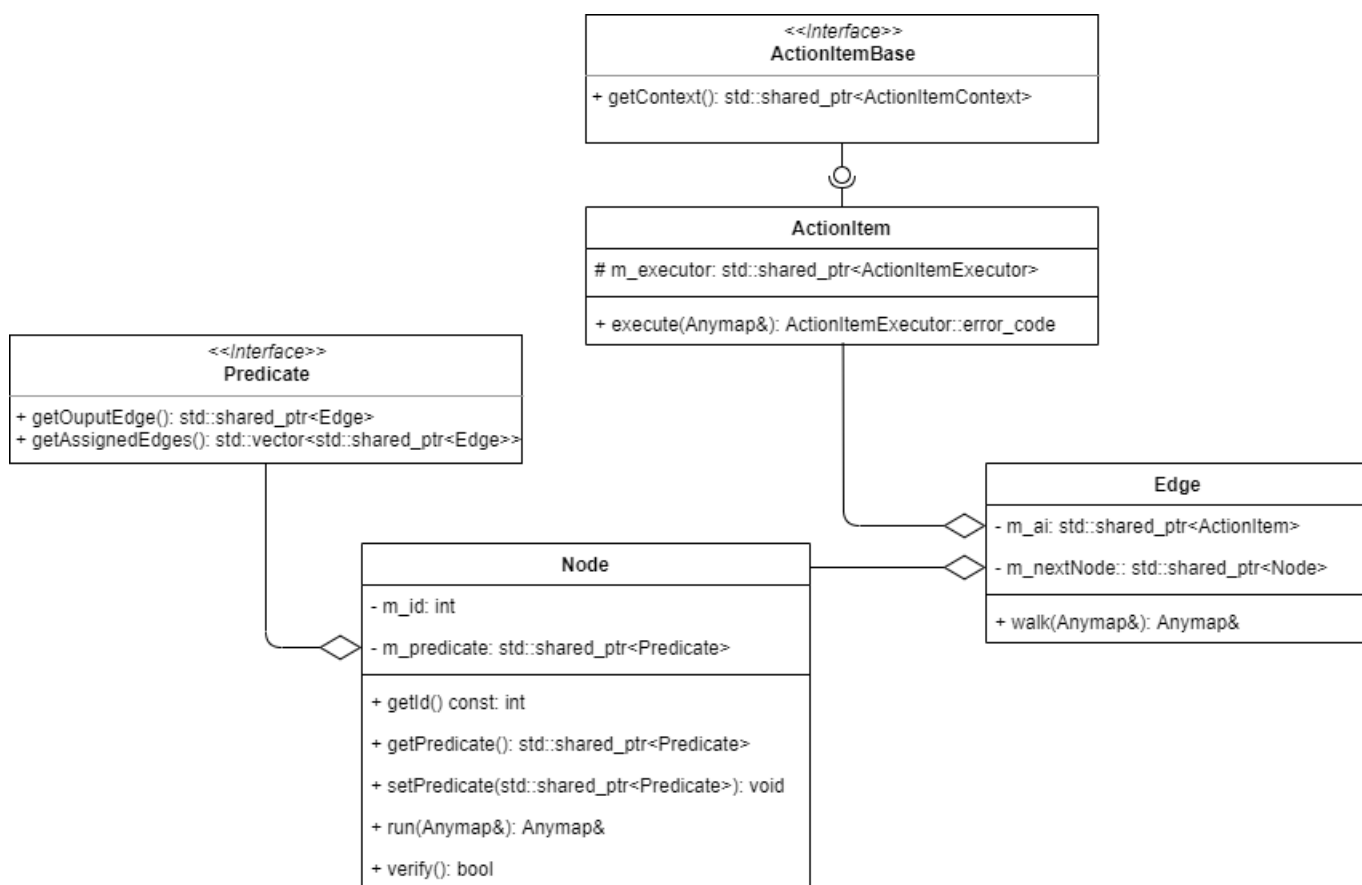


Рисунок 3. Текущая структура классов, связанная с графовыми моделями в comsdk

В существующей структуре классов можно выделить следующие недостатки:

- 1) Отсутствует класс графа, который давал бы удобный интерфейс графовым моделям.
- 2) Отсутствует контейнер, который бы инкапсулировал все узлы, относящиеся к конкретной графовой модели.
- 3) Индекс узла графа задаётся пользователем при инициализации, что не гарантирует его уникальности.
- 4) Отсутствует контейнер, который бы инкапсулировал все рёбра, относящиеся к конкретной графовой модели.
- 5) Отсутствует объект, который бы описывал связи между узлами и рёбрами; вместо этого эти связи прописаны в самих узлах и рёбрах, что затрудняет операции с графовой моделью (преобразования и проч.).
- 6) Функции-предикаты привязываются к узлам, а не к рёбрам, что не соответствует новым требованиям.

- 7) В текущей версии задачей функций-предикатов фактически является отбор рёбер, которые должны быть выполнены, а не проверка соответствия данных в узле определённому формату.

Таким образом, процесс разработки новой структуры графового модуля библиотеки `comsdk` должен быть направлен как на выполнение требований, описанных в разделе 2.1, так и на устранение недостатков текущей версии, описанных выше.

3 Архитектура программной реализации

Проектирование новой программной архитектуры графового модуля библиотеки comsdk проводилось с учётом доступа разработчика к средствам стандартной библиотеки языка C++ стандарта C++-11 и библиотеки шаблонных классов Standart Template Library (STL).

3.1 Узлы и рёбра графа

Структура разработанных классов узла графа (Node) и ребра (Edge) представлена на рисунках 4 и 5 соответственно.

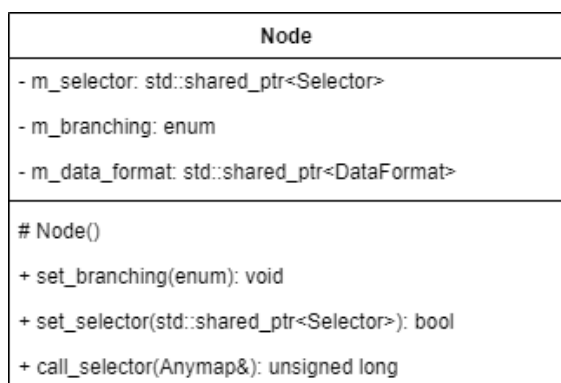


Рисунок 4. Класс узла графа

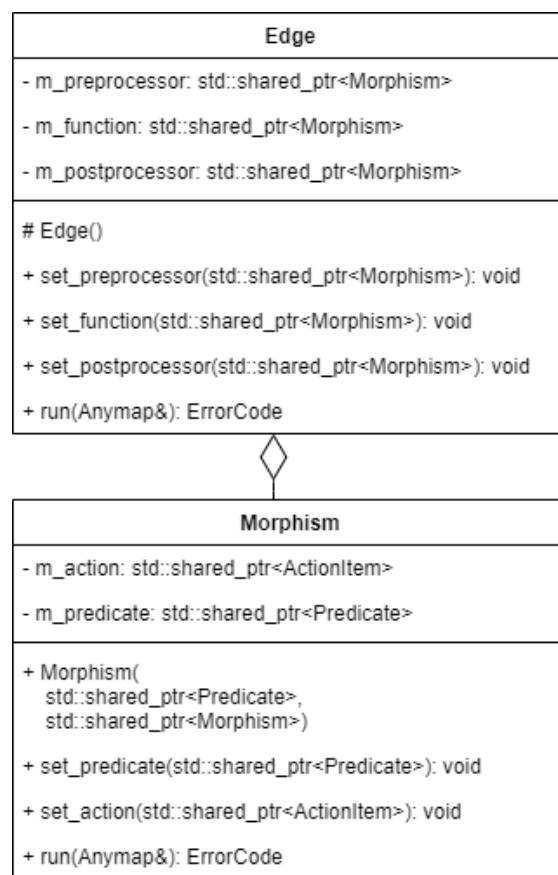


Рисунок 5. Класс ребра графа

Класс узла содержит в себе объект, описывающий состояние данных **DataFormat**, указатель на объект класса **Selector** и тип стратегии параллельного выполнения исходящих из него рёбер. Кроме того класс **Node** предоставляет интерфейс для назначения и вызова селектора и назначения стратегии распараллеливания. В свою очередь класс **Selector** является обёрткой над функтором, который по входным данным создаёт массив логических переменных, где 1 на i -той позиции означает, что при обходе необходимо выполнить i -тое ребро, выходящее из узла. Класс ребра предоставляет интерфейс для назначения ему до трёх морфизмов и их выполнения, что в полной мере соответствует обозначенным в разделе 2.1 требованиям.

3.2 Хранение узлов и рёбер графа

Задача хранения данных узлов и рёбер была отведена непосредственно классу графа (Graph). Поскольку основной операцией с узлами и рёбрами является обращение к ним по их индексу, в качестве контейнеров для них были выбраны динамические массивы (`std::vector`). Кроме того, данному классу была отведена задача хранить данные о топологии графовой модели в виде матрицы смежности. Данное решение позволяет легко проверять наличие ребра между двумя узлами и, кроме того, даёт возможность сразу узнать индекс ребра при его существовании. Поскольку граф должен формироваться засчёт поочерёдного добавления в него узлов и рёбер, в качестве контейнера для этой матрицы был взят двумерный динамический массив. UML-диаграмма данного класса приведена на рисунке 6.

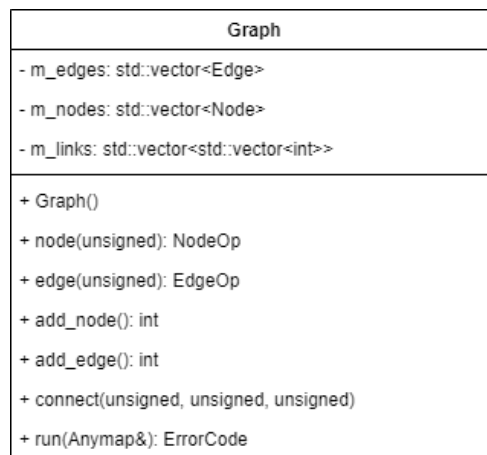


Рисунок 6. Класс графа

Помимо этого для удобства доступа к структуре графа во время обхода были спроектированы две вспомогательные структуры данных, позволяющих отделить данные о топологии графа от отдельно взятых узлов и рёбер. UML-диаграммы этих структур представлены на рисунке 7.

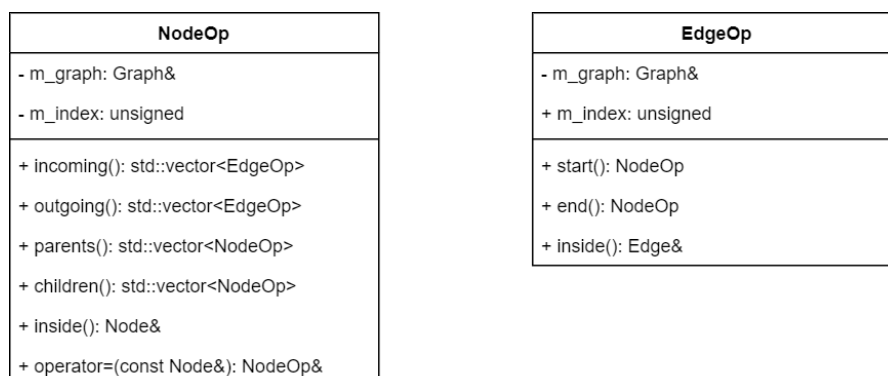


Рисунок 7. Дополнительные структуры данных

Например, класс **NodeOp** предоставляет интерфейс для получения входящих и выходящих рёбер для конкретного узла, соседних узлов и при необходимости позволяет обратиться непосредственно к интерфейсу самого узла, при этом внутри объекта этого класса хранится

только ссылка на граф и индекс узла, что позволяет эффективно использовать память, когда необходимости обращаться к интерфейсу самого узла нет. Аналогично в классе `EdgeOp` хранится только индекс ребра и ссылка на граф. Интерфейс данного класса позволяет получить начальную и конечную вершины для данного ребра и при необходимости обратиться к интерфейсу самого ребра (`Edge`).

Итоговая UML-диаграмма классов, спроектированных в рамках графового модуля `comsdk` представлена на рисунке 8.

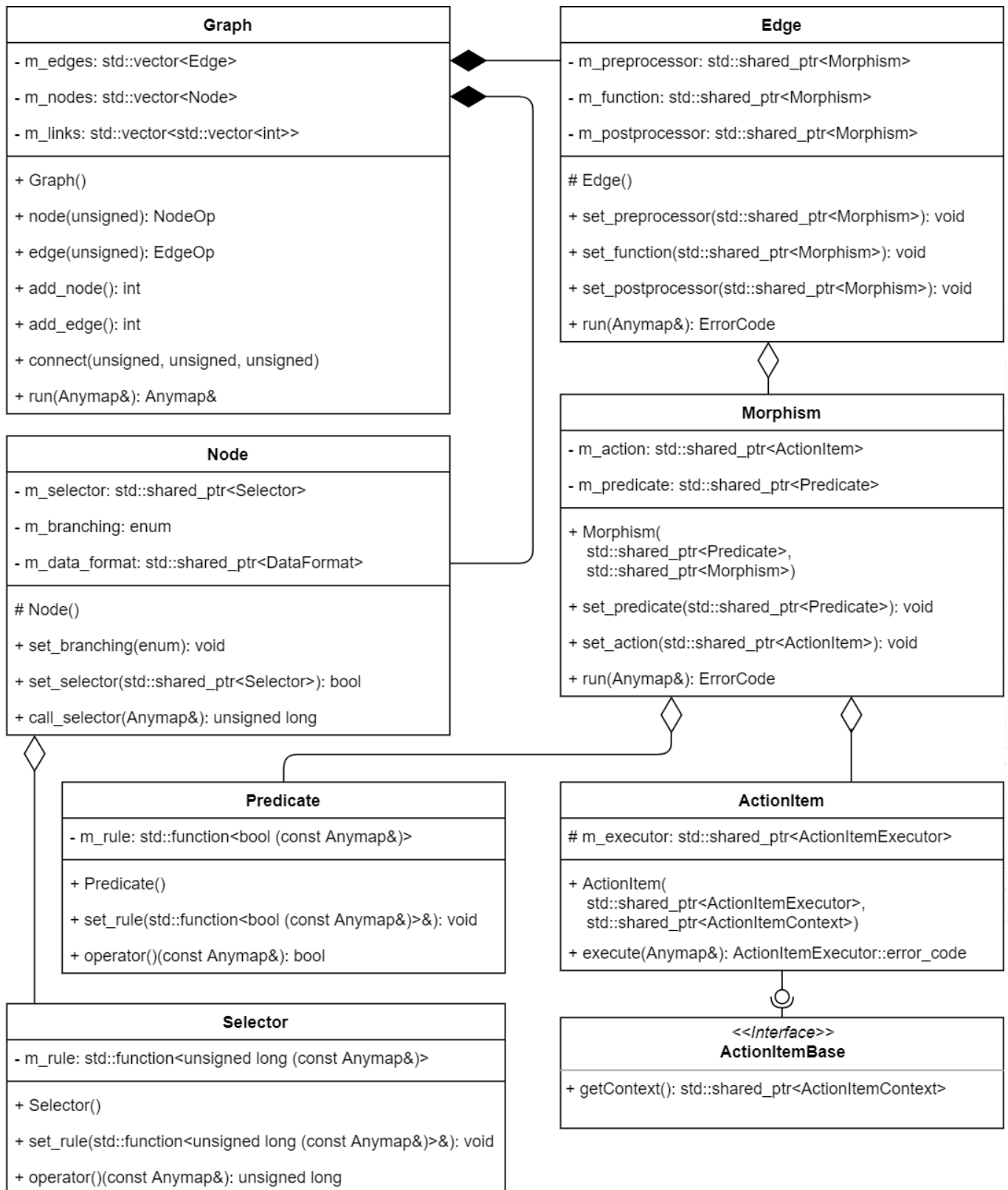


Рисунок 8. Предлагаемая структура классов

ЗАКЛЮЧЕНИЕ


В результате выполнения данного курсового проекта была изучена концепция графо-ориентированного подхода к решению задач проектирования и проведено сравнение двух различных реализаций данного подхода (pSeven и GBSE). Была изучена структура программного комплекса GBSE и используемый в нём формат описания графовых моделей aDot. Был изучен исходный код модуля библиотеки языка C++ comsdk, отвечающего за формирование графовых моделей. Была разработана новая структура классов для этого модуля, которая должна устранить недостатки существующей реализации, которые также были выделены в ходе выполнения курсового проекта.

Список использованных источников

- 1 Соколов А.П. Першин А.Ю. Описание формата данных aDOT (advanced DOT). 2020.
- 2 Соколов А.П. Описание формата данных aINI (advanced INI) [Электронный ресурс]. Облачный сервис SA2 Systems. [Офиц. сайт]. 2020. URL: <https://sa2systems.ru/nextcloud/index.php/f/403527>.
- 3 В.О. Голубев. Разработка web-приложений, реализующих бизнес-логику работы пользователя в САПР на основе графоориентированной методологии. 2020.
- 4 Соколов А.П. Першин А.Ю. Система автоматизированного проектирования композиционных материалов. Часть 1. Концепции, архитектура и платформа разработки // Известия СПбГЭТУ «ЛЭТИ». 2020. № 8-9.
- 5 Соколов А.П. Голубев В.О. Система автоматизированного проектирования композиционных материалов. Часть 3. Графоориентированная методология разработки средств взаимодействия пользователь–система // Известия СПбГЭТУ «ЛЭТИ». 2021. № 2/2021. С. 43–57.
- 6 Соколов А.П. Першин А.Ю. Графоориентированный программный каркас для реализации сложных вычислительных методов // Программирование. 2018. № X.
- 7 PRADIS. Общее описание системы [Оф. документация]. 2007.
- 8 PRADIS. Методы формирования и численного расчёта математических моделей переходных процессов [Оф. документация]. 2007.
- 9 Alexey M. Nazarenko Alexander A. Prokhorov. Hierarchical Dataflow Model with Automated File Management for Engineering and Scientific Applications // Procedia Computer Science. 2015. Т. 66. URL: <https://www.sciencedirect.com/science/article/pii/S1877050915034055?pes=vor>.
- 10 Соколов А.П. Першин А.Ю. Программный инструментарий для создания подсистем ввода данных при разработке систем инженерного анализа // Программная инженерия. 2017. Т. 8, № 12. С. 543–555.
- 11 Condition - pSeven 6.31.1 User Manual [Электронный ресурс] [Офиц. сайт]. 2022. (дата обращения 07.03.2022). URL: <https://www.datadvance.net/product/pseven/manual/6.31.1/blocks/Condition.html>.
- 12 Расчётные схемы - Руководство пользователя pSeven 6.27 [Электронный ресурс] [Офиц. сайт]. 2021. Дата обращения: 15.11.2021. URL: <https://www.datadvance.net/product/pseven/manual/ru/6.27/workflow.html#workflow-links>.
- 13 DOT Language | Graphviz [Электронный ресурс] [Офиц. сайт]. 2022. (дата обращения 07.03.2022). URL: <https://www.graphviz.org/doc/info/lang.html>.

Выходные данные

Тришин И.В.. Проектирование структуры графовых моделей в программном комплексе GBSE по дисциплине «Модели и методы анализа проектных решений». [Электронный ресурс] — Москва: 2021. — 26 с. URL: <https://sa2systems.ru:88> (система контроля версий кафедры РК6)

Постановка:  канд. физ.-мат. наук, Соколов А.П.

Решение и вёрстка:  студент группы РК6-71Б, Тришин И.В.

2021, осенний семестр

”