МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«НОВОСИБИРСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ» (НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ, НГУ)

КАФЕДРА СИСТЕМ ИНФОРМАТИКИ

Тимур Александрович Золотухин

Методы и средства навигации при интерактивной визуализации структурированной информации

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

по направлению высшего профессионального образования

230100.68 ИНФОРМАТИКА И ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

Тема диссертации утверждена распоряжением по НГУ №___ от «___»_____200__г.

Тема диссертации скорректирована распоряжением по НГУ №___ от «___»_____200__г.

Руководитель

Касьянов Виктор Николаевич

Доктор физико-математических наук, профессор

Новосибирск, 2013г.

Содержание

ВВЕДЕНИЕ

Визуализация информации играет большую роль в жизни человека. Считается, что около 90% всей получаемой информации человек получает за счет зрения. Человечество за тысячи лет преодолело путь от простейшей визуализации в виде наскальных рисунков до карт, схем и диаграмм. В настоящее время визуализация – неотъемлемый элемент обработки сложной информации о строении объектов.

С другой стороны многие структуры данных, представляющие практический интерес в математике и информатики, могут быть представлены в виде графов. Одним из основных классов являются иерархические и/или атрибутированные графы[1, 2].

Преимущества графов во многих случаях становятся ощутимыми только при наличии хороших инструментальных средств их визуализации и обработки. Поэтому в настоящее время, в мире, происходит значительный рост интереса к методам и средствам визуализации графов, о чем свидетельствует рост публикаций, содержащих описание новых алгоритмов и способов визуализации графов, а так же их реализации в системах.

Главный принцип визуализации информации - один рисунок может заменить тысячу слов.

Магистерская работа выполняется при содействии специалистов из компании Интел, занимающихся оптимизацией кодогенератора C/C++ компилятора. Данное сотрудничество осуществляется в рамках проекта по созданию системы для визуализации иерархических атрибутированных графов.

Целью магистерской работы является построение кроссплатформенной расширяемой системы визуализации атрибутированных иерархических графов, создание удобной навигации по ним, а также предоставление дополнительных средств, позволяющих повысить качество сопровождения, разрабатываемой системы.

Новизна работы состоит в предоставлении уникальных средств навигации по графам, а так же в гибкой расширяемости системы, чего не предоставляют программы-аналоги.

В ходе развития компиляторов было придумано несколько структур, которые представляются в виде графов: синтаксические деревья, графы потоков управления и графы вызовов. Каждый из этих графов имеет свое практическое применение. Так

синтаксические деревья используются при построении внутреннего представления программы в компиляторах или интерпретаторах. Графы потока управления - для оптимизаций в компиляторах и утилитах статического анализа кода, а графы вызовов для отладки программ. Все эти графы объединяет, то, что их можно рассматривать как атрибутированные иерархические графы.

Представление этих графов в разных компиляторах может быть разным, причем нет гарантии, что оно не будет отличаться в разных версиях одного и того же компилятора.

Для преодоления данной проблемы, необходимо либо, чтобы сам компилятор переводил граф из своего внутреннего представления в файл, одного из поддерживаемых системой Visual Graph форматов, либо чтобы это делала вспомогательная программа.

После того как граф будет сохранен в файл, система Visual Graph сможет прочитать его оттуда, визуализировать, и предоставить пользователю средства навигации по нему.

При разработке системы использовалась следующая литература:

1. Графы в программировании: обработка, визуализация и применение. Авторы: В.Н. Косьянов и В.А. Евстигнеев. Книга содержит изложение фундаментальных основ современных компьютерных технологий, связанных с применением теории графов. Приведены основные модели, методы и алгоритмы прикладной теории графов. Рассмотрены задачи рисования графов и визуальной обработки графовых моделей. Описаны области приложения, такие как хранение и поиск информации, трансляция и оптимизация программ, анализ, преобразование и распараллеливание программ, параллельная и распределенная обработка информации. Данная книга помогла при изучении предметной области, а так же при проектировании системы (в основном при проектировании хранения данных). Именно из этой книги были взяты основные понятия, касающиеся Графовых моделей.
2. Приемы объектно-ориентированного проектирования. Паттерны проектирования. Авторы: Э. Гамма, Р.Хелм, Р. Джонсон, Дж. Влисседс. В данной книге описываются простые и изящные решения типичных задач, возникающих в объектно-ориентированном проектировании. Паттерны появились потому, что многие разработчики искали пути повышения гибкости и степени повторного использования своих программ. Найденные решения воплощены в краткой и легко применимой на практике форме. Данная книга помогла при проектировании системы.

ГЛАВА 1 ПОСТАНОВКА ЗАДАЧИ

Для достижения поставленной цели необходимо решить следующие задачи:

1. изучение предметной области, которое включает в себя ознакомление с существующей на сегодняшний день терминологией, методами визуализации графов, программными средствами для визуализации графов, с существующими средствами навигации по графам, а также изучение программ-аналогов (см. раздел 2.2);
2. сбор требований к системе, который включает в себя получение технического задания (далее ТЗ) от Заказчика, а также замечаний по тестовой эксплуатации им предыдущей версии Программы;
3. реинжиниринг существующей архитектуры системы, с целью удовлетворения новым требованиям;
4. реинжиниринг хранения графов в системе, с целью обеспечения быстрой загрузки графов из Входных файлов, предоставления возможности работать с графами больших размеров, а так же осуществления быстрого поиска интересующей пользователя информации;
5. реинжиниринг существующих средств навигации по графам, а так же разработка и реализация новых. В частности разработка и реализация инструмента позволяющего проводить сравнительный анализ двух графов
6. разработка и реализация библиотеки, позволяющей сохранять в нужном формате промежуточное представление транслируемой программы в компиляторе Заказчика.

ГЛАВА 2 ОБЗОР ПРОГРАММНЫХ СРЕДСТВ

В данной главе будет проведен обзор программных средств, которые использовались для решения вышеописанных задач (см. Глава 1). Так же в этой главе будут рассмотрены программные средства, которые являются аналогами Программы.

2.1 Используемые программные средства

В данном разделе будет проведен анализ используемых средств. Будут описаны их общие плюсы и минусы, а так же плюсы и минусы в контексте поставленной задачи.

2.1.1 Java

Java - объектно-ориентированный язык программирования, разработанный компанией Sun Microsystems. Приложения Java обычно компилируются в специальный байт-код, поэтому они могут работать на любой виртуальной Java-машине (JVM) независимо от компьютерной архитектуры [6].

Общие достоинства:

- кроссплатформенность;
- простота разработки;
- бесплатная лицензия.

Общие недостатки:

- так как Программы, написанные на Java, исполняются на виртуальной Java-машине, то их производительность ниже, чем у программ, исполняющихся непосредственно на операционной системе.

Достоинства в контексте Программы:

- кроссплатформенность. Одним из пунктов ТЗ, которое было выдано Заказчиком, значится пункт, в котором написано, что Программа должна работать под операционными системами: Windows, Linux, Mac OS. Поэтому Java подходит для этих целей как нельзя лучше.

Недостатки в контексте Программы:

- отсутствуют.

2.1.2 SQLite

SQLite - легковесная встраиваемая реляционная база данных. Исходный код библиотеки передан в общественное достояние[7].

Общие достоинства:

- кроссплатформенность;
- не требует дополнительной установки;
- является общественным достоянием;
- имеет одни из наиболее лучших показателей поиска и записи среди встраиваемых баз данных.

Общие недостатки:

- отсутствуют

Достоинства в контексте Программы:

- скорость поиска и записи элементов;
- кроссплатформенность. Есть поддержка тех операционных систем, которые необходимы Заказчику.

Недостатки в контексте Программы:

- отсутствуют.

В качестве библиотеки для работы с SQLite была взята sqlite4java[8], т.к. она имеет более хорошие технические показатели, чем обычная SQLiteJDBC для Java.

2.1.3 JGraph

JGraph – активно разрабатываемая библиотека для визуализации графов[9].

Общие достоинства:

- BSD лицензия;
- простота в освоении;
- развивающийся проект;
- предоставляет несколько базовых раскладчиков графов;
- предоставляет широкий спектр средств для визуализации вершин и ребер графа.

Общие недостатки:

- не слишком высокая скорость отображения. Это связано с тем, что данный продукт использует Java Swing;
- малое количество опций для настройки существующих раскладчиков.

Достоинства в контексте Программы:

- написан на Java и использует Swing. Т.е. это небольшой минус по скорости, но зато большой плюс по встраиваемости в систему.

Недостатки в контексте Программы:

- из-за малого количества опций для настройки существующих раскладчиков есть необходимость в написании патчей к коду для исправления поведения некоторых раскладчиков.

Использование данного продукта так же обосновано требованием Заказчика.

2.1.4 GraphML

GraphML — язык описания графов на основе XML [10].

Общие достоинства:

- простота в использовании. Т.к. данный формат основан на использовании XML, то можно использовать готовые решения для создания и чтения XML файлов. К тому же, при необходимости, пользователь всегда может подправить интересующие его вещи в файле руками;
- позволяет хранить ориентированные, неориентированные и смешанные графы;
- позволяет хранить иерархические атрибутированные графы.

Общие недостатки:

- файлы, в которых хранятся Графовые модели, занимают большой объем дискового пространства.

Достоинства в контексте Программы:

- простота в использовании;
- есть возможность хранить Графовые модели.

Недостатки в контексте Программы:

- отсутствуют

Использование данного формата обосновано требованием Заказчика.

2.1.5 DOT

DOT (Graphviz формат) — это язык описания графов в виде простого текста [11].

Общие достоинства:

- компактнее, по сравнению с GraphML, при этом оставаясь легким для человеческого восприятия;
- позволяет хранить ориентированные, неориентированные и смешанные графы;
- позволяет хранить иерархические атрибутированные графы.

Общие недостатки:

- предоставляет меньше возможностей по сравнению с GraphML, так в частности не предоставляет возможности пользователю описывать и добавлять собственные параметры к ребрам и вершинам.

Достоинства в контексте Программы:

- простота в использовании;
- есть возможность хранить Графовые модели.

Недостатки в контексте Программы:

- отсутствуют.

Использование данного формата обосновано требованием Заказчика.

2.1.6 Substance

Substance – это библиотека стилей для пользовательского интерфейса, написанного с использованием библиотеки Swing. В Java библиотеке Swing имеется возможность изменения отображения графических компонентов. В базовой комплектации библиотеки содержится всего несколько стилей, которые не блещут своей красотой. В библиотеке Substance содержится множество проработанных стилей на любой вкус [12].

Общие достоинства:

- простота в использовании;

- имеет BSD лицензию;
- содержит множество проработанных стилей, что дает хороший внешний вид пользовательского интерфейса на любой вкус.

Общие недостатки:

- отсутствуют.

Достоинства в контексте Программы:

- легко встраивается в систему.

Недостатки в контексте Программы:

- отсутствуют.

2.1.7 Maven

Maven — фреймворк для автоматизации сборки проектов, специфицированных на XML-языке [13].

Общие достоинства:

- бесплатная лицензия;
- интеграция с Java;
- переносимость проекта, собираемого данным фреймворком под все популярные интегрированные среды разработки.

Общие недостатки:

- сложность в изучении;
- есть открытые библиотеки, которые собираются не Maven и попадают в центральный репозиторий позже, чем выйдет официальный релиз.

Достоинства в контексте Программы:

- легко встраивается в систему.

Недостатки в контексте Программы:

- отсутствуют.

2.1.8 JUnit

JUnit — библиотека, предназначенная для модульного тестирования программного обеспечения, написанного на языке Java [14].

Общие достоинства:

- простота в использовании;
- бесплатная лицензия;
- интеграция с Java.

Общие недостатки:

- не позволяет выполнять запланированные цепочки тестов в нужном порядке

Достоинства в контексте Программы:

- легко встраивается в систему

Недостатки в контексте Программы:

- отсутствуют

2.2 Обзор аналогичных программ

В настоящее время на рынке представлен достаточно широкий круг систем для работы с графами. Наиболее известными являются: aiSee[15], yEd[16] и cytoscape[17].

Область применения, выбранных выше систем, частично совпадает с областью применения Программы. Стоит пояснить слово “частично”. В данном контексте оно означает, что их область применения шире, т.к. многие из них рассчитаны на редактирование существующих графов и создание новых. Программа ассоциирует себя исключительно с просмотром уже существующих графов без возможности их редактирования.

Основными задачами, подобных систем являются отображение графов и инструменты навигации по ним. Поэтому в рамках этих двух задач и будем рассматривать особенности каждой из систем.

Вначале несколько слов о каждой из рассматриваемых систем.

aiSee - платная система, которая автоматически рисует раскладку графа, описанного

на языке GDL (см. Рис. 1:). Затем пользователь может интерактивно исследовать данный граф (без возможности редактирования), отпечатать его и сохранить в различных форматах.

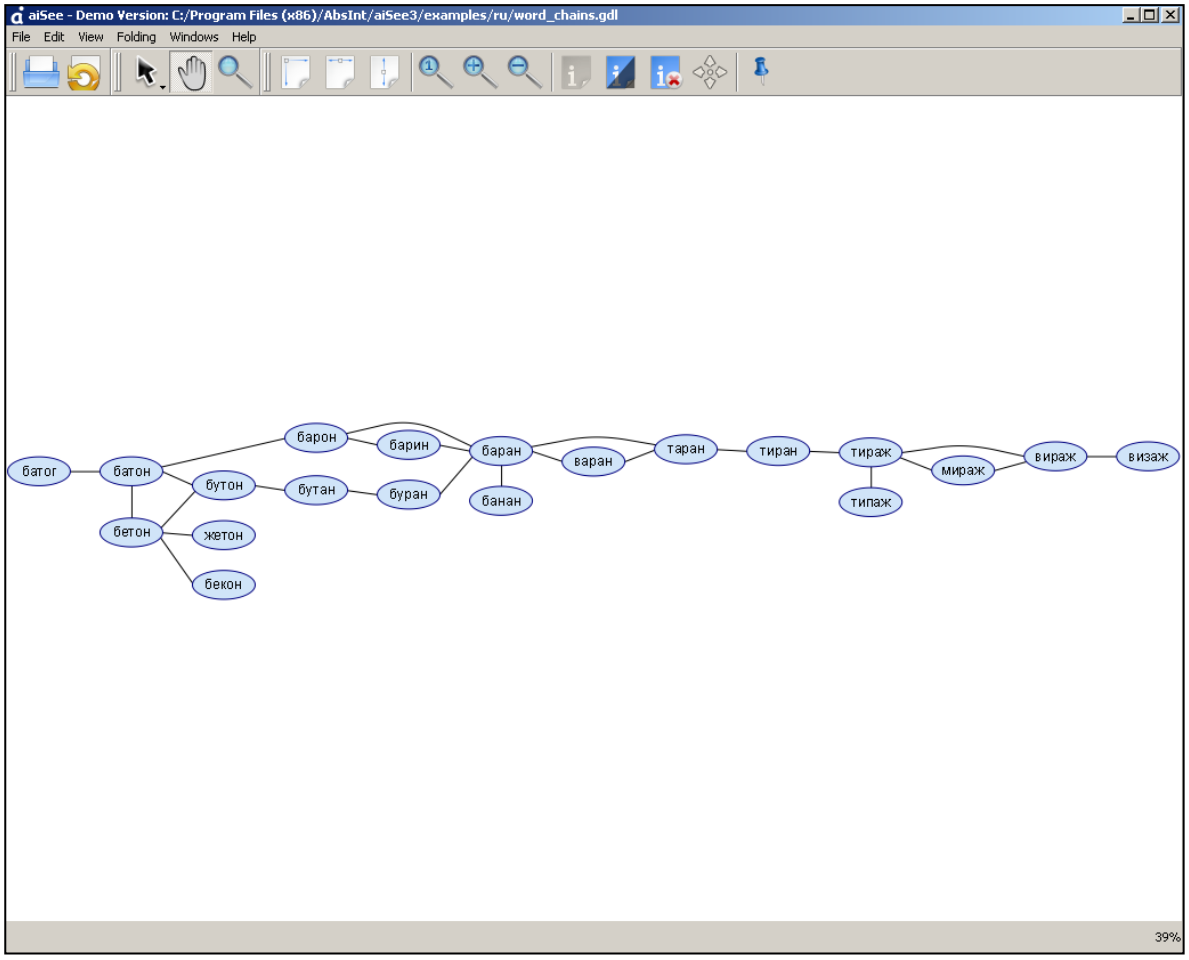

Рис. 1: Скриншот системы aiSee

Первоначально aiSee был разработан для визуализации структур данных, обрабатываемых компиляторами. На сегодняшний день его используют десятки тысяч людей по всему миру в самых различных областях, в том числе:

- бизнес-менеджмент (структурные схемы предприятий, визуализация бизнес-процессов),
- генеалогия (семейные деревья),
- разработка программного обеспечения (блок-схемы, графы потока управления, графы вызовов функций),
- анализ объёма стека и так далее.

Cytoscape – свободная система с открытым исходным кодом для визуализации и

анализа сетей (см. Рис. 2:).

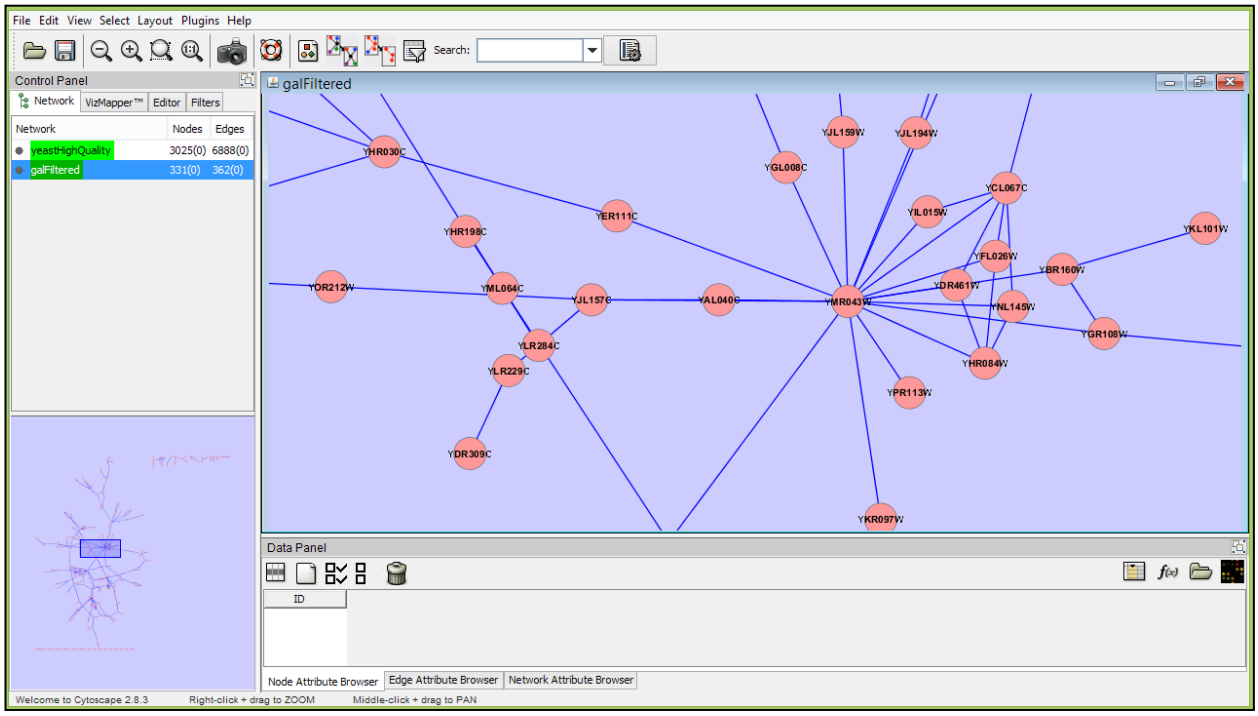

Рис. 2: Скриншот системы Cytoscape

Основная область применения данной системы является биоинформатика. Данная система стала очень популярной благодаря тому, что легко расширяется, позволяя сторонним разработчикам писать для нее всевозможные плагины.

yEd – система, которая может быть использована для быстрого создания и редактирования высококачественных диаграмм (см. Рис. 3:).

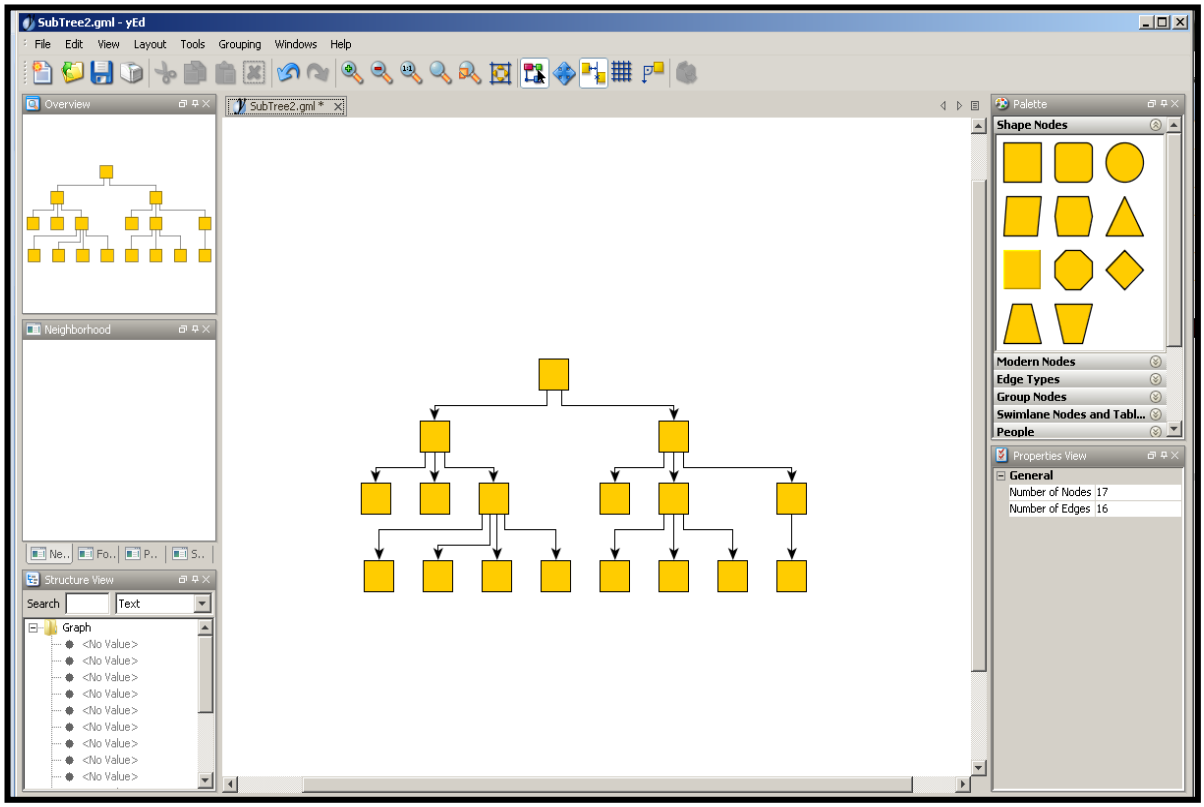

Рис. 3: Скриншот системы yEd

Создание диаграмм происходит вручную или с помощью импортирования из внешних данных. После чего к полученной диаграмме можно применить богатый набор алгоритмов для проведения анализа с последующим получением необходимой информации.

Каждая из рассматриваемых систем, так или иначе, отображает графы и имеет свои методы для управления отображением этих графов.

В aiSee присутствует несколько раскладчиков, и множество настроек для них, которые в незначительной степени влияют на результат. Качество раскладки определенных типов графов очень высокое.

В yEd присутствует большое количество раскладчиков и опций для них, которые предоставляют возможность тонкой настройки визуализации каждого графа пользователем.

Cytoscape имеет как собственные алгоритмы раскладки, так и сторонние, среди которых можно выделить присутствие алгоритмов, которые используются в yEd. Стоит так же отметить, что результаты, получаемые в yEd намного лучше, чем аналогичные результаты в Cytoscape с настройками раскладчиков по умолчанию.

Программа предоставляет несколько раскладчиков, главным из которых является иерархический раскладчик, который был максимально адаптирован для работы с графами, используемыми в компиляторах. Но при желании раскладчик может быть изменен и/или

настроен под другие типы задач.

Закончив с задачей отображения графов, мы можем приступить к рассмотрению особенностей средств навигации.

Выделим основные средства навигации, которые предоставляет система aiSee для решения данной задачи:

- Рабочий стол - инструмент, который визуализирует графовую модель целиком, выдавая пользователю статичную картинку, которую нельзя изменять, передвигая элементы, например.
- Поисковик - инструмент, позволяющий пользователю искать элементы графовой модели с помощью их имен. Он поддерживает задание регулярных выражений, выбор категорий элементов для поиска и сохранение предыдущих поисковых запросов.

Выделим основные средства навигации, которые предоставляет система yEd для решения данной задачи:

- Рабочий стол схожий с рабочим столом, который предоставляет система aiSee. Но в отличие от aiSee:
    a. есть возможность работать с несколькими графами одновременно в одном экземпляре программы (создается вкладка для каждого графа);
    b. есть возможность редактирования элементов графа, начиная от смены их положения и размеров до задания атрибутов, влияющих на их визуализацию.
- Навигатор - инструмент, отображающий граф, находящийся в текущей вкладке.
- Миникарта - инструмент, показывающий весь граф, находящийся в текущей вкладке целиком.

Выделим основные средства навигации, которые предоставляет система Cytoscape для решения данной задачи:

- Рабочий стол, схожий с рабочим столом, который предоставляет система yEd. Отличие заключается в том, что система Cytoscape не умеет работать с иерархическими графами.
- Миникарта, идентичная миникарте в системе yEd.

- Атрибутная панель - инструмент, позволяющий отображать текущие атрибуты и задавать новые. Данный инструмент выглядит в виде таблицы, по горизонтали которой расположены имена атрибутов, по вертикали список выделенных вершин, а в ячейках соответствующие значения того или иного атрибута для той или иной вершины.
- Фильтр - инструмент, позволяющий осуществлять поиск вершин и ребер по заданным условиям на атрибутах.

Выделим основные средства навигации, которые предоставляет система Visual Graph (подробное описание всех ниже перечисленных инструментов можно найти в разделе 3.2.2):

- Рабочий стол, визуализирующий части графовой модели. К особенностям данного инструмента можно отнести:
    a. возможность влиять на визуализацию элементов по средствам задания определенных атрибутов;
    b. возможность работать с несколькими графами одновременно, в одном экземпляре программы;
    c. возможность редактирования элементов графа (изменение положения элементов и их размеров без возможности сохранения);
    d. возможность просмотра интересующей пользователя области графа в другой вкладке. Это одна из важнейших особенностей данного инструмента. Человек устроен таким образом, что для понимания чего-то сложного, ему нужно это сложное разделить на множество маленьких простых кусочков, которые он сможет легко проанализировать;
    e. возможность визуализации пользовательских атрибутов.
- Навигатор, схожий с навигатором, который предоставляет система yEd. Основным отличием является то, что в системе yEd навигатор отображает граф, находящийся в текущей вкладке, а не все графы, с которыми работает пользователь, как это сделано в Visual Graph.
- Миникарта, схожая с миникартой в системе yEd.
- Атрибутная панель, отображающая набор атрибутов у выделенных элементов

графовой модели, а так же позволяющая управлять их визуализацией на рабочем столе.

- Фильтр и поисковая панель, осуществляют поиск элементов по заданному выражению. К особенностям данного инструмента можно отнести:
    a. возможность задания сложных булевских выражений;
    b. использование типов атрибутов при задании выражения
- Блокнот. Несмотря на то, что данный инструмент не обладает большинством функций, являющихся достаточно стандартными для большинства современных блокнотов, таких как подсветка синтаксиса, поиск, с использованием регулярных выражений и многое другое. Тем не менее, его функциональности вполне хватает для решения тех задач, которые встают перед пользователем Программы, а именно:
    a. возможность открывать множество текстовых файлов и связать их с графами, для последующего перехода из элемента графа к фрагменту в текстовом файле;
    b. подсветка результатов поиска.
- Средства структурного анализа, такие как поиск циклов и кратчайших путей, а так же сравнительный анализ двух графов. Сложно охарактеризовать плюсы и минусы, данных инструментов, можно только сказать, что без подобных инструментов обойтись практически невозможно. Чтобы это понять, необходимо просто представить, сколько времени пользователь потратит, чтобы глазами сравнить два графа содержащих порядка десяти вершин, а после этого представить, что количество вершин может быть порядка сотни и даже тысячи вершин.

ГЛАВА 3 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

Особенностью Программы является ее высокая степень модульности и расширяемости. Программный продукт выполнен в виде ядра системы и набора Плагинов. Это позволяет в значительной мере упростить процесс добавления в систему дополнительной функциональности: вычислительные алгоритмы, навигационные средства и т.п.

3.1 Этап проектирования

В ходе проектирования системы можно выделить следующие этапы:

- изучение предметной области;
- сбор требований к системе;
- реинжиниринг существующей архитектуры системы;
- реинжиниринг существующей архитектуры базы данных;
- разработка алгоритма сравнения графов.

Рассмотрим подробнее каждый из них.

3.1.1 Изучение предметной области

В рамках изучения предметной области была проведена следующая работа: ознакомление с существующей на сегодняшний день терминологией, методами визуализации Графовых моделей, программными средствами для визуализации графов, с существующими средствами навигации по Графовым моделям, а также изучение программ-аналогов (см. раздел 2.2).

Введем наиболее важные понятия, которые будут в дальнейшем использоваться в тексте дипломной работы.

3.1.1.1 Иерархический атрибутированный граф

Пусть *G* обозначает граф произвольного вида, элементы (вершины и ребра) которого отличаются один от другого какими-либо пометками, называемыми их именами.

Граф *C* называется *фрагментом* графа *G*, если *C* часть графа *G*, т.е. подмножество элементов графа *G*.

F – иерархия фрагментов графа *G*, если *F* – такое множество фрагментов графа *G*, что *G* принадлежит *F* и для любых двух фрагментов C_1 и C_2 из *F* либо фрагменты C_1 и C_2 не пересекаются, либо один из них является частью (*подфрагментом*) другого. Фрагмент *G* – основной (главный) фрагмент иерархии *F*. Фрагмент *C*, принадлежащий *F*, – элементарный, если в *F* нет фрагментов *G*, являющихся подфрагментами фрагмента *C*.

Пусть задана некоторая иерархия фрагментов F графа G. Для любых C_1 и C_2, принадлежащих *F*, фрагмент C_1 – прямой подфрагмент C_2 (или, что тоже самое, фрагмент, *непосредственно вложенный* в C_2), если C_1 – подфрагмент C_2 и не существует такого C_3 принадлежащего *F*, отличного от C_1 и C_2, что C_1 вложен в C_3, а C_3 в свою очередь вложен в C_2.

Иерархический граф H=(G,T) состоит из графа *G* и корневого дерева *T*, вершины которого соответствуют элементам некоторой иерархии в *G*, а дуги отражают отношение их непосредственной вложенности. *T* называется *деревом вложенности*, а *G – основным графом* иерархического графа *H* [2].

Атрибутированный граф – это граф вида (G, A_v, A_e), где *G=(V,E)* – граф, A_v (A_e) – функция отображающая множество вершин (ребер) во множество атрибутов. Под атрибутом понимается пара: (имя атрибута, значение атрибута).

Иерархический граф *H=(G,T)* называется иерархическим атрибутированный графом (далее Графовая модель), если G является атрибутированным графом.

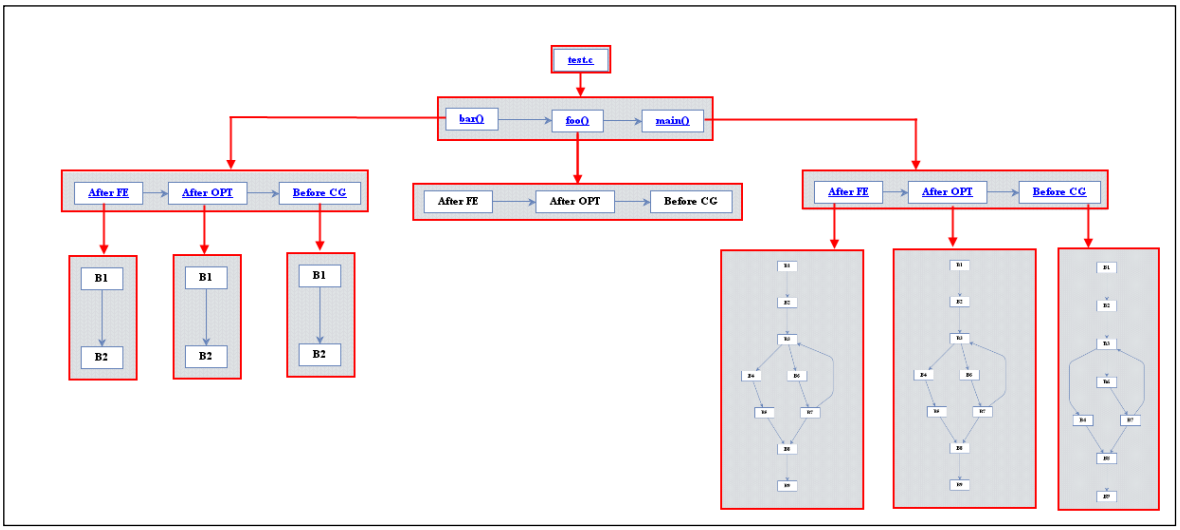

Рис. 4: Пример иерархического атрибутированного графа

3.1.2 Сбор системных требований

Сбор системных требований включает в себя:

1. получение ТЗ от Заказчика и согласование с ним новой функциональности

Программы;

2. получение замечаний о существующей функциональности предыдущей версии Программы, которые отметил Заказчик в ходе тестовой эксплуатации.

В результате сбора системных требований было установлено, что Заказчик хочет, чтобы Программа обладала следующей функциональностью:

- открытие Графовых моделей, хранящихся в формате graphml[10], graphviz[11];
- визуализация интересующих пользователя частей Графовой модели;
- предоставление средств поиска интересующей пользователя информации в Графовых моделях;
- сравнение двух графов и выделение наибольшего общего подграфа;
- возможность связывать файлы, содержащие дополнительную информацию, с Графовыми моделями, а затем осуществлять переход из Графовых моделей к интересующей информации, хранящейся в этих файлах;
- управление визуализацией атрибутов в выбранной части Графовой модели;
- возможность расширять Программу.

Так же Заказчик хочет видеть качественное сопровождение системы, которое включает в себя:

- библиотеку, которая могла бы сохранять графы, возникающие в системе Заказчика, во Входном формате для Программы.
- руководство для пользователя и разработчика;
- набор тестовых сценариев для тестирования программных модулей, а так же тестирования пользовательского интерфейса;
- документация к проекту.

3.1.3 Архитектура системы

Сложность разрабатываемой системы, а так же необходимость длительной поддержки системы требуют серьезного и аккуратного подхода к построению архитектуры программы.

Все модули программы разбиты на два класса:

- Первый класс называется ядром системы, а модули, входящие в него,

называются сервисами. Сервисы не предоставляют напрямую никакой функциональности для пользователя. Их основной целью является обслуживание модулей, принадлежащих ко второму классу, которые в свою очередь нацелены на работу с пользователем.

- Второй класс состоит из модулей, которые называются Плагинами. Эти модули ориентированы на работу с пользователем, но могут предоставлять функциональность и другим Плагинам. Стоит отметить, что все Плагины в системе абсолютно независимы и напрямую не знают о существовании друг друга.

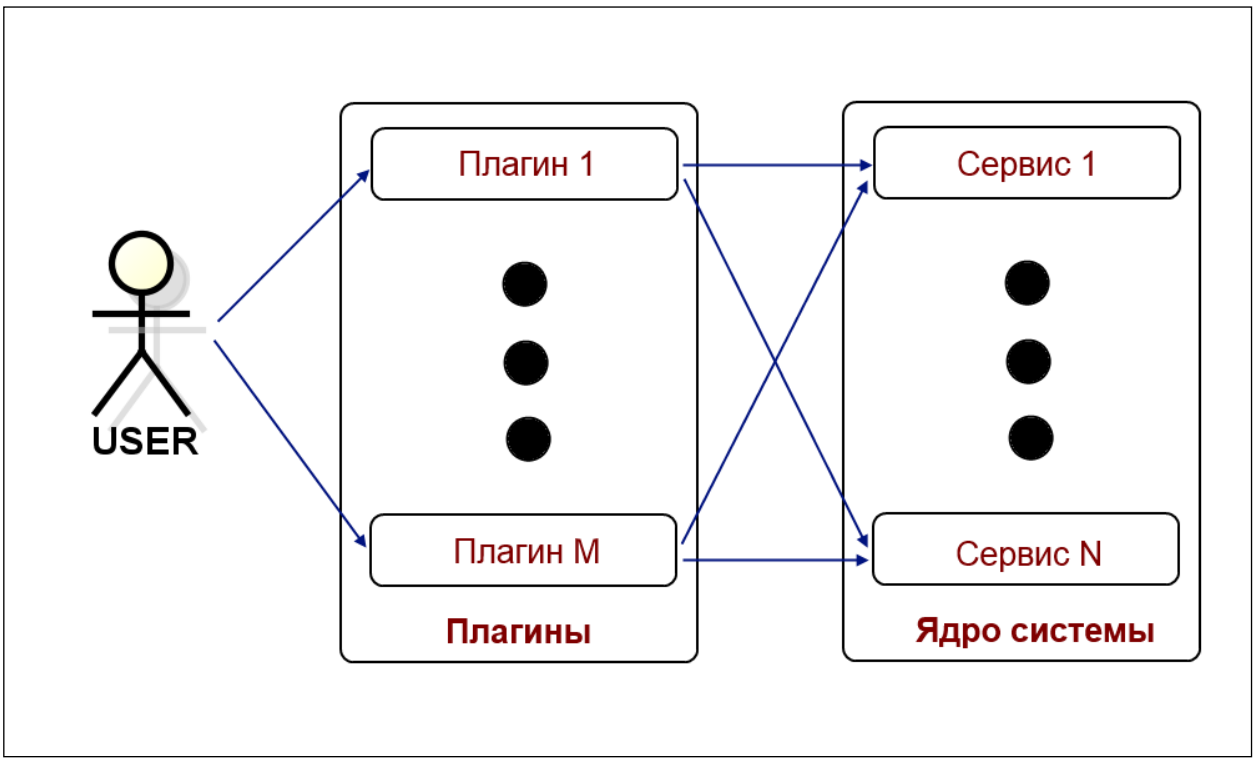

Рис. 5: Иллюстрация архитектуры

3.1.4 Алгоритм поиска максимального общего подграфа двух графов

Как известно алгоритм поиска изоморфизма двух графов, как и поиск максимального общего подграфа двух графов, принадлежит классу NP и решается полным перебором с применением эвристик для уменьшения числа вариантов для перебора. Алгоритм, разрабатываемый для Программы, идет по тому же пути и работает следующим образом:

1. на вход принимается два атрибутированных (возможно и без атрибутов) графа;
2. строится двудольный граф, вершинами, которого являются вершины первого и второго графа (первая доля и вторая доля, соответственно). Далее от каждой вершины одной доли проводится ребро к вершине другой доли с весом равным

проценту совпадения этих двух вершин (от нуля до единицы соответственно). Где нуль означает кардинальное несовпадение вершин, а единица, соответственно, наоборот их полное совпадение;

3. к полученному двудольному графу применяется венгерский алгоритм решения задачи о назначениях для получения максимального паросочетания максимального веса [4, 5]. Очевидно, что такое паросочетание может быть не в единственном числе и далее остается перебрать все эти варианты.

Не смотря на то, что в конце алгоритм сводиться к перебору, данная эвристика позволяет отсеять большое количество неверных решений, тем самым уменьшив общее время работы алгоритма. Происходит это главным образом за счет того, что графы, с которыми работает пользователь в Программе в большинстве своем атрибутированные, и пользователю разрешено выставлять вес того или иного атрибута, который будет использован при подсчете веса ребра в двудольном графе, описанном выше.

3.1.5 Хранение данных

Размер Входного графа может достигать сотни мегабайт. Следовательно, хранение такого количество информации в оперативной памяти компьютера может привести к тому, что она быстро закончится. Одно из решений этой проблемы – это кэширование данных на жесткий диск, но в связи с тем, что перед проектом стояла задача навигации, которая подразумевает под собой также выборку всевозможной информации из Графовых моделей, то было принято решение об использовании реляционной базы данных.

В качестве реляционной базы данных была выбрана встраиваемая база данных SQLite[7]. В отличие от большинства популярных реляционных баз данных, для SQLite не требуется установка сервера, а вся клиент-серверная архитектура сводится к работе с файлами.

На Рис. 6: представлена архитектура базы данных, которая используется Программой для хранения Графовых моделей. База данных с такой архитектурой позволяет Графовой модели иметь более одного корневого графа, в отличие от предыдущей версии Программы.

Так же стоит отметить, что было добавлено кэширование элементов Графовой модели в оперативную память, а так же общей структуры Графовой модели, благодаря чему общая производительность Программы значительно возросла.

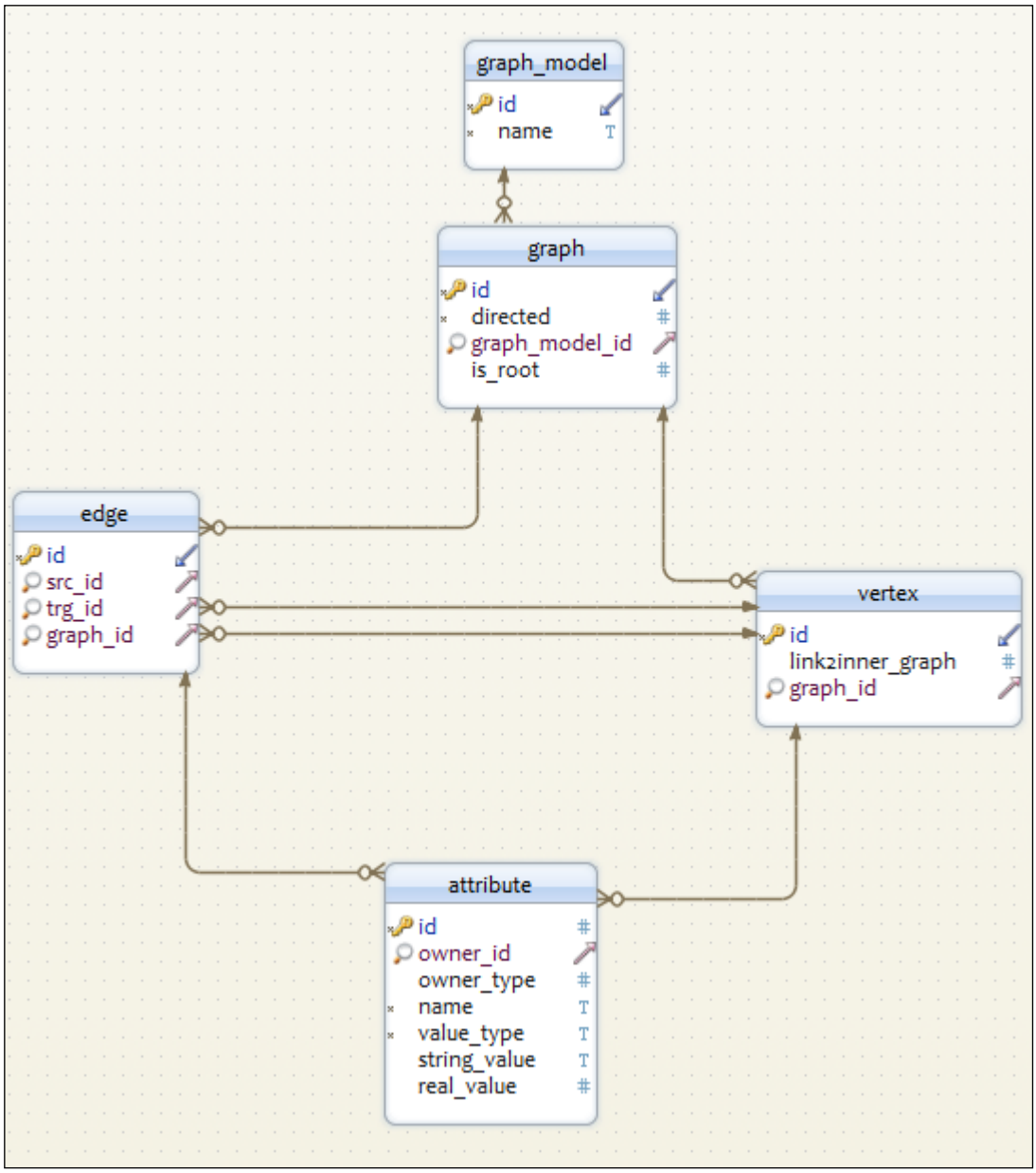

Рис. 6: Структура базы данных для хранения Графовых моделей.

3.2 Этап реализации

В ходе разработки Программы можно выделить следующие этапы:

- Реализация сервисов, образующих ядро системы. Главными, из которых являются: сервис хранения Графовых моделей и сервис, отвечающий за расширяемость системы.
- Реализация средств навигации по Графовым моделям.

Рассмотрим подробнее каждый из этих пунктов.

3.2.1 Реализация ядра системы

Все сервисы инициализируются при старте Программы и направлены на то, чтобы облегчить труд разработчика, сократить его код, а так же стандартизовать некоторые типичные операции.

Основные сервисы:

1. Сервис хранения Графовых моделей в Программе. Как видно из названия данный сервис занимается складированием Графовых моделей и выдачей их целиком или по частям. Особенностью его реализации является то, что он дает возможность загружать Графовую модель по частям, что позволяет реализовать загрузчик, не хранящий Входной граф целиком в оперативной памяти. Для хранения Графовых моделей используется встраиваемая реляционная база данных SQLite (структура которой описана в разделе 3.1.4).
2. Сервис управления Плагинами. Как видно из названия данный сервис занимается добавлением плагинов в одну из точек расширения, а так же поддерживает связь между ними, рассылая запросы и уведомления.

   В Программе существует три точки расширения, в которые можно добавить Плагин (см. Рис. 7:):

   1. инструментальная панель,
   2. меню,
   3. одна из панелей (западная, центральная, южная или юго-западная).

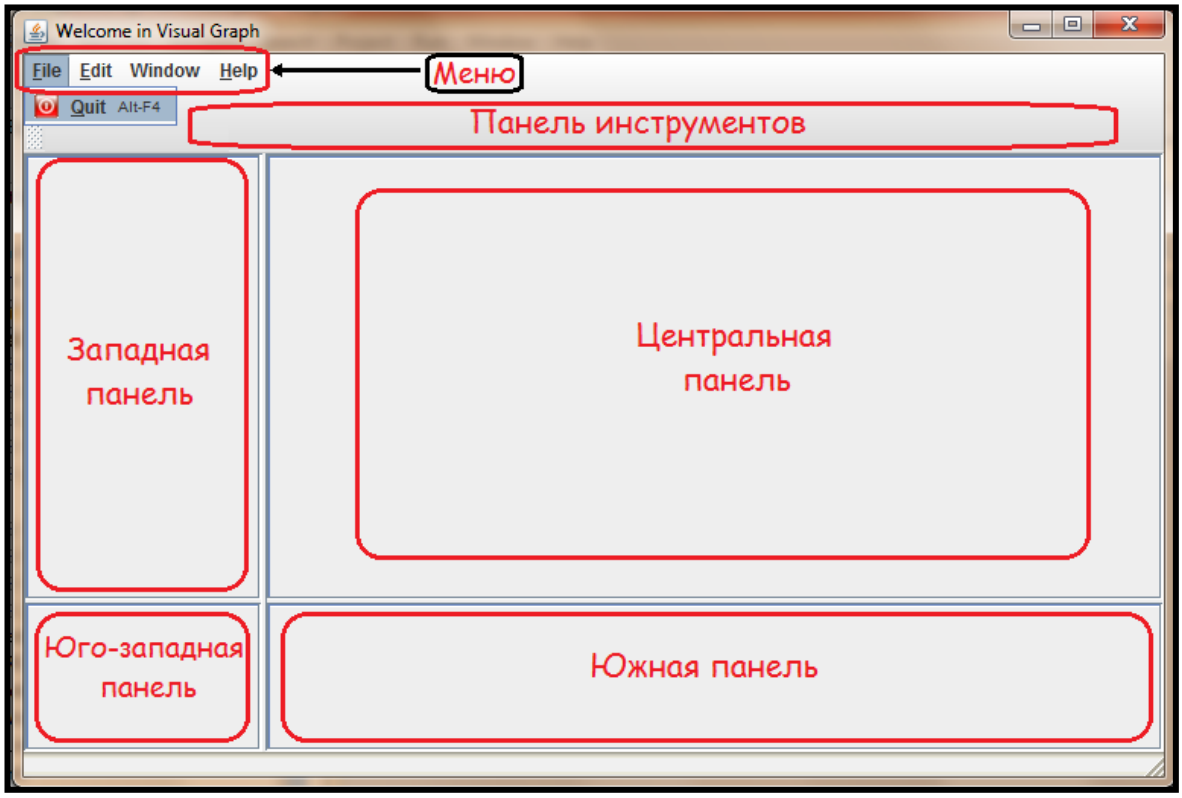

Рис. 7: Программа Visual Graph без Плагинов. На данном рисунке показаны точки расширения

Как было сказано выше, Плагины изолированы друг от друга и могут общаться между собой за счет системы уведомлений и запросов. Т.е. каждый Плагин может отправить запрос или уведомление данному сервису и тот в свою очередь разошлет его по всем Плагинам. Так же Плагин, исполняющий чей-то запрос, может отослать заказчику (Плагину, породившему этот запрос) уведомление о состоянии его запроса. Разработчику Плагина необходимо только прописать реакцию своего Плагина на те, или иные запросы и уведомления.

Вспомогательные сервисы:

- Сервис логирования позволяет разработчику Плагина вызвать из своего кода и записать интересующую его информацию или ошибку. В дальнейшем данная информация может помочь при отладке Программы.
- Сервис, хранящий конфигурацию Программы, позволяет разработчику Плагина сохранить настройки его Плагина, а так же запросить интересующую его информацию. Например, размер окна Плагина, который был в прошлый раз установлен пользователем.
- Сервис, выдающий оконные сообщения. В отличие от сервиса логирования,

который просто записывает какую-то информацию, данный модуль визуально предупреждает пользователя о том, что что-то произошло. Например, ошибка при открытии файла с Графовой моделью.

3.2.2 Реализация средств навигации по Графовым моделям

Программа предоставляет следующий набор инструментов (см. Рис. 8:) для навигации по Графовым моделям:

1. рабочий стол;
2. миникарта для графа, находящегося в текущей вкладке;
3. навигатор, визуализирующий Графовые модели в виде дерева;
4. атрибутная панель, позволяющая выбрать атрибуты, которые будут визуализироваться в выбранных пользователем элементах графа в рабочей области;

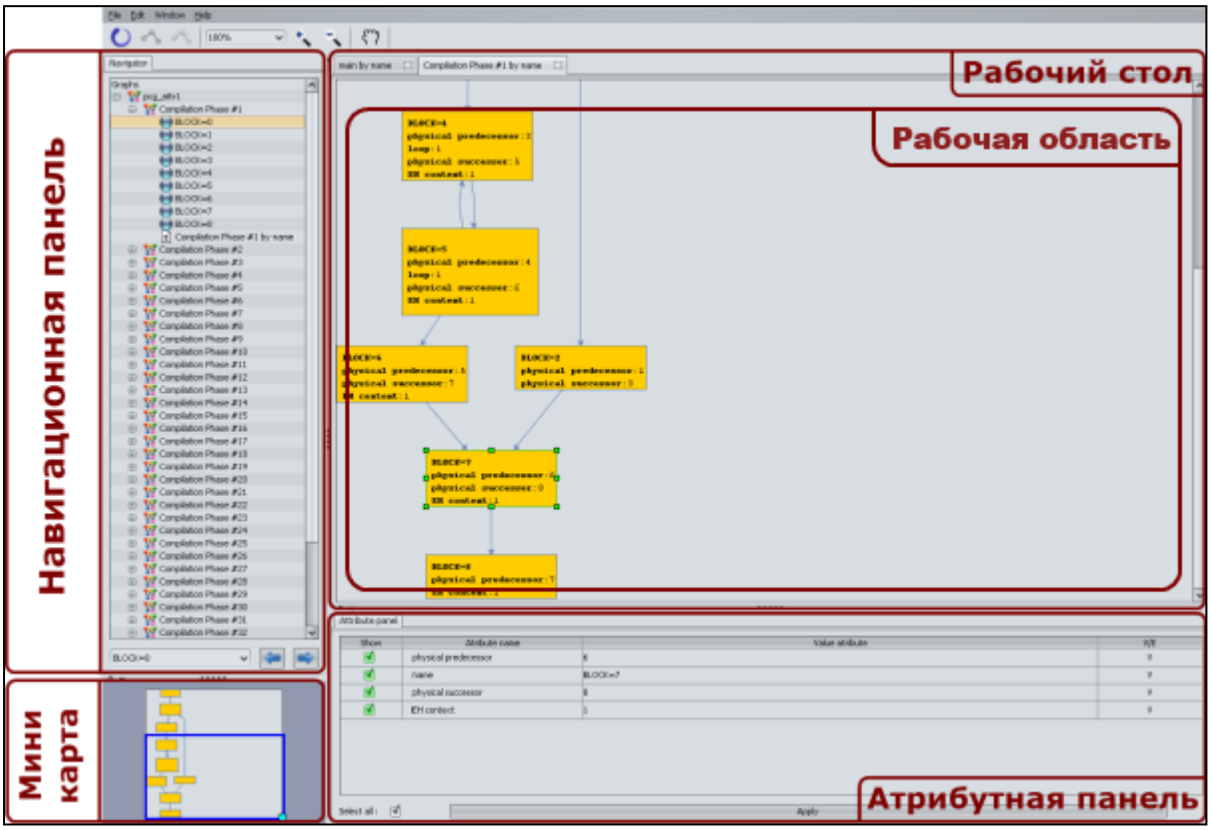

Рис. 8: Пользовательский интерфейс системы Visual Graph

5. фильтр, осуществляющий поиск вершин и ребер по заданным условиям в текущей вкладке;
6. поисковая панель, осуществляющая поиск вершин по заданным условиям во всей Графовой модели, а так же в какой - либо из ее частей;
7. блокнот, позволяющий загружать файлы с дополнительной информацией и

связывать их с Графовыми моделями.

Стоит отметить, что все эти модули реализованы в виде Плагинов (как и вся функциональность Программы). Теперь рассмотрим каждый из модулей подробнее.

3.2.2.1 Рабочий стол

Рабочий стол – инструмент, который состоит из набора вкладок, которые открывает пользователь для визуализации выбранной части Графовой модели. Каждая вкладка состоит из рабочей и не рабочей областей. Рабочая область – это область, которую пользователь видит непосредственно в данный момент и в которой визуализируются вершины и ребра графа, а также атрибуты и их значения, связанные с этими вершинами и ребрами. Не рабочая область – область, которую пользователь не видит, но может туда попасть, используя скроллинг.

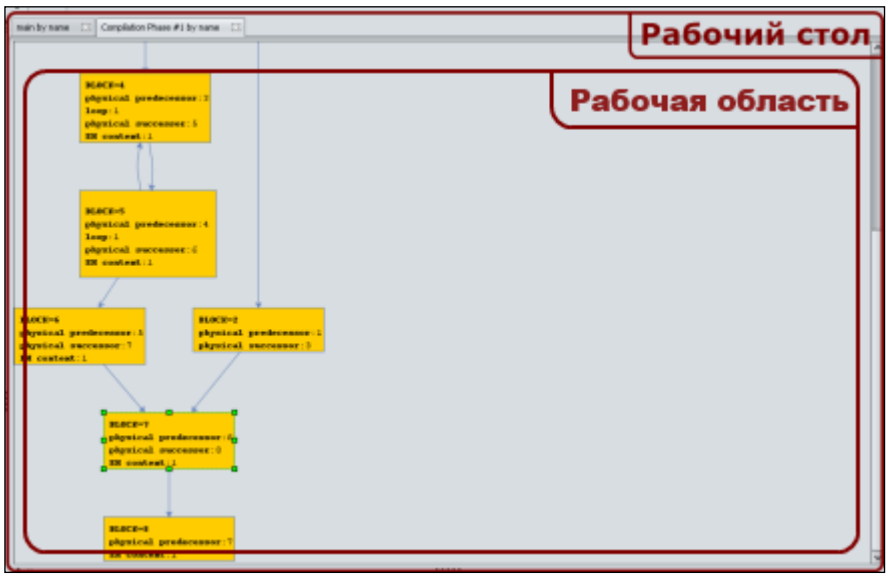

Рис. 9: Рабочий стол

3.2.2.2 Миникарта

Миникарта – инструмент, который позволяет увидеть весь граф целиком, находящейся в текущей вкладке, а также увидеть текущую рабочую область.

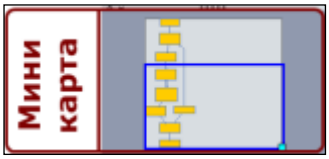

Рис. 10: Миникарта

3.2.2.3 Навигатор

Навигатор – инструмент, визуализирующий графы, с которыми работает пользователь, в виде дерева. В данном дереве отображены только вершины, т.к. ребра несут меньшую

смысловую информацию, но очень сильно засоряют рисунок. Для быстрого поиска по дереву реализована строка поиска, которая позволяет пользователю без труда найти интересующую его вершину по имени. Так же из навигатора есть доступ к визуализации интересующих пользователя вершин графовой модели с помощью рабочего стола. Т.е. пользователь может выделить интересующую его вершину или группу вершин и открыть их в новой вкладке. При этом будут достроены все ребра между выбранными вершинами.

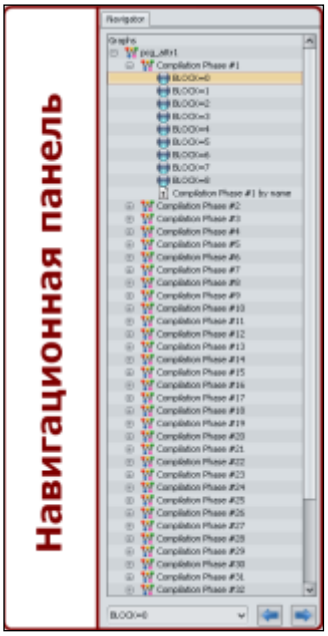

Рис. 11: Навигатор

3.2.2.4 Атрибутная панель

Атрибутная панель - инструмент, который позволяет управлять визуализацией атрибутов для выбранных вершин и ребер в текущей вкладке. Для этого пользователю необходимо выбрать интересующие его вершины и ребра в текущей вкладке на рабочем столе, после чего отметить в атрибутной панели галочками те атрибуты, которые он хочет визуализировать.

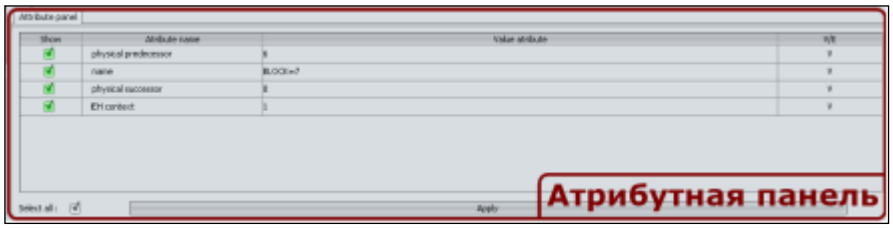

Рис. 12: Атрибутная панель

3.2.2.5 Фильтр

Фильтр – инструмент, который использует тот факт, что пользователь работает с атрибутированными графами. Данный инструмент осуществляет поиск вершин и ребер по заданным условиям в текущей вкладке. Построение условий осуществляется за счет

атрибутов и их значений. На Рис. 13:, показан механизм задания выражения, состоящего из логических связок, скобок и условий двух видов:

1. строковые условия вида: имя атрибута = подстрока;
2. числовые условия вида: первое значение ≤ имя атрибута ≤ второе значение.

После того как пользователь закончит формирование выражения, оно исполнится на элементах графа, находящихся в текущей вкладке.

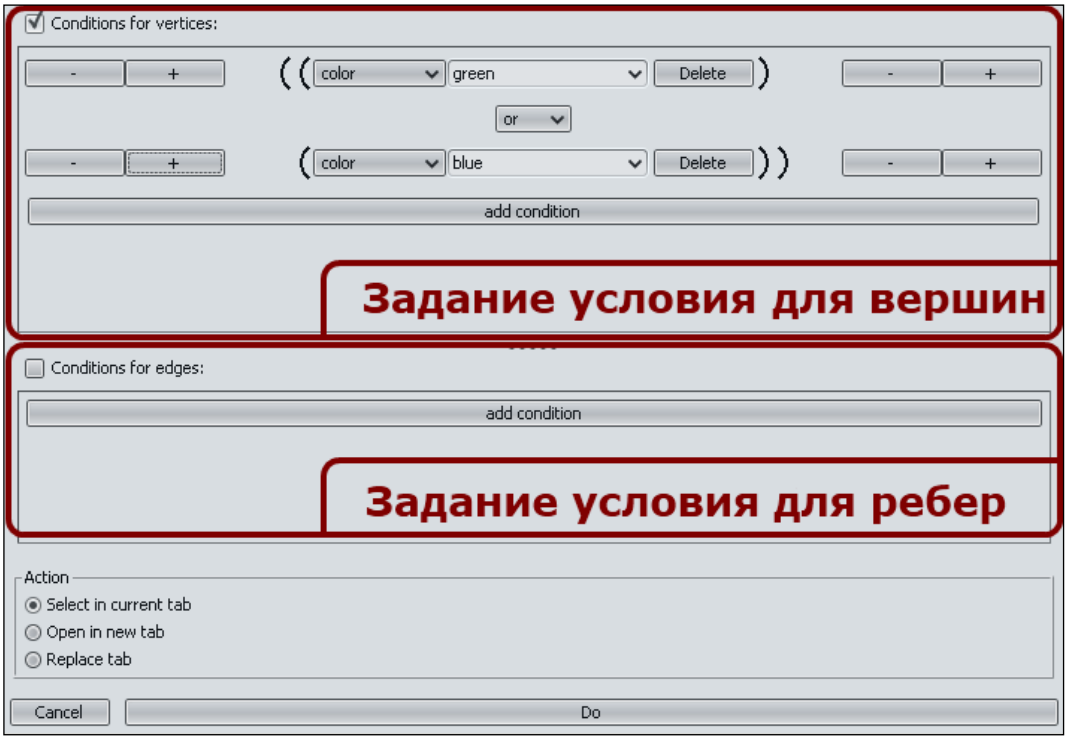

Рис. 13: Фильтр

В результате чего пользователь получит набор вершин и ребер, которые будут удовлетворять заданным им условиям (см. Рис. 14:).

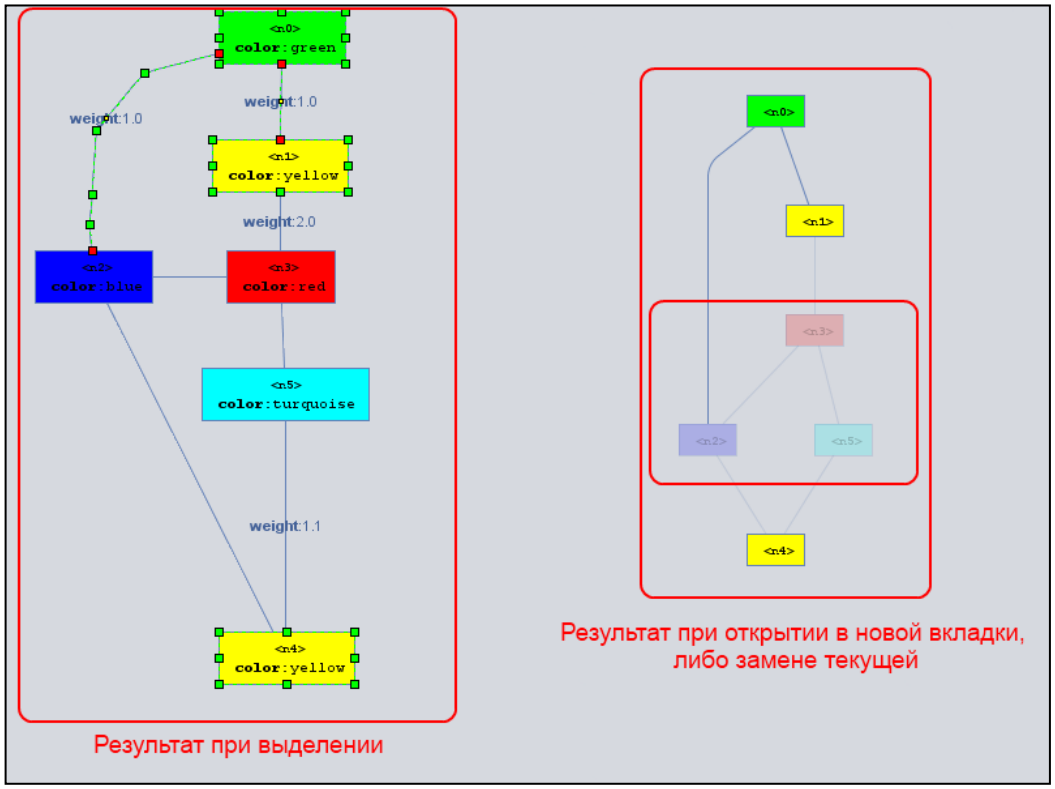

Рис. 14: Результат работы фильтра

3.2.2.6 Поисковая панель

Поисковая панель – инструмент, который, как и фильтр, использует тот факт, что пользователь работает с атрибутированным графом. В отличии от фильтра, данный инструмент позволяет задавать условия только для вершин и осуществляет поиск не только для графа в текущей вкладке, но и для всех его внутренних графов и их внутренних графов и т.д. по иерархии вниз. Результатом такого поиска является результирующее дерево (см. Рис. 15:), в котором красными крестиками отмечены вершины, не удовлетворяющие заданному условию.

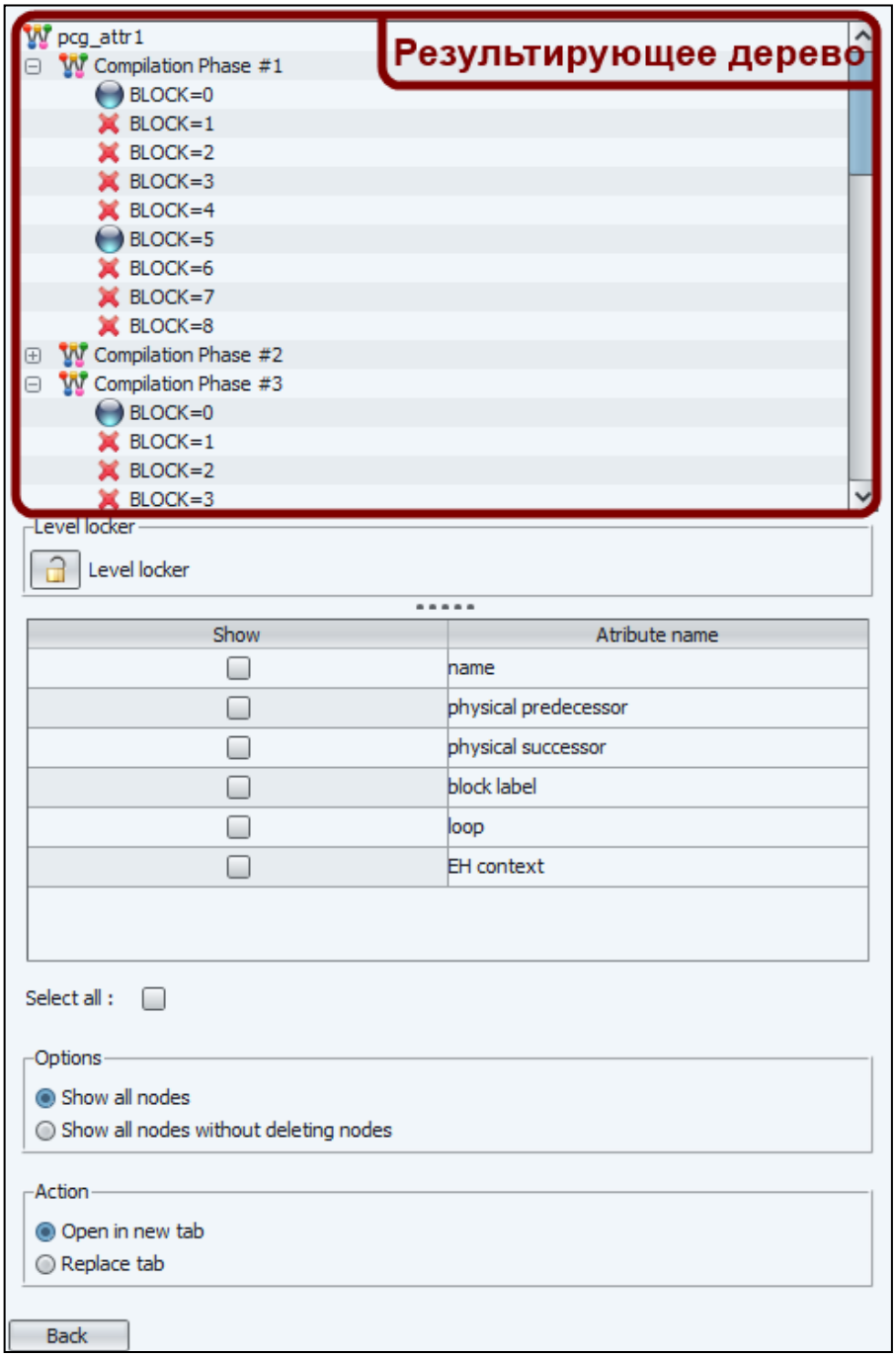

Рис. 15: Результат работы поисковой панели

Как и в случае с навигатором пользователь может выделить интересующую его вершину или группу вершин и открыть их в новой вкладке. При этом будут достроены все ребра между выбранными вершинами.

3.2.2.7 Инструмент поиска максимального подграфа двух графов

Результат действия данного инструмента, а так же сам алгоритм описан в разделе 3.1.4. В данном разделе речь пойдет о его свойствах с точки зрения пользователя.

Есть два пути использования данного инструмента:

1. выделить два графа в навигаторе и в контекстном меню вызвать функцию сравнения двух графов;

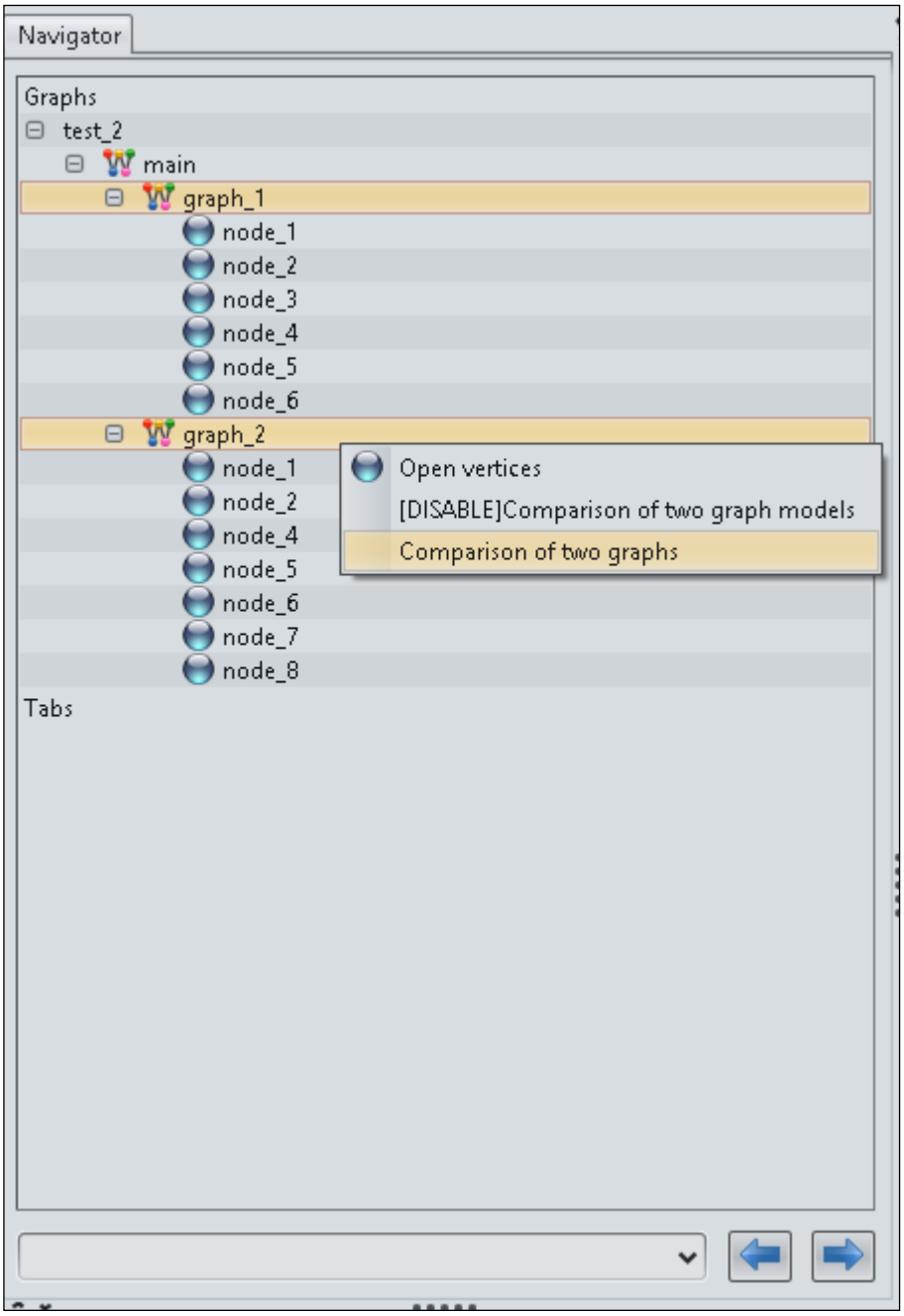

Рис. 16: Вызов сравнения двух графов из навигатора

2. выделить две вершины в текущей вкладке на рабочем столе (описание в разделе 3.2.2.1), которые содержат внутренний граф, и в контекстном меню вызвать функцию сравнения двух графов.

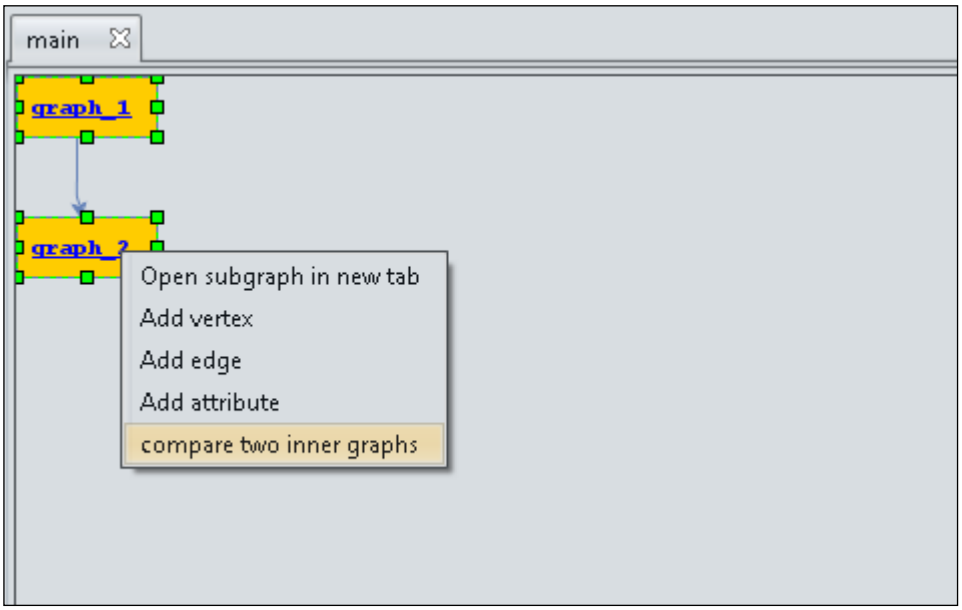

Рис. 17: Вызов сравнения двух графов из текущей вкладки на рабочем столе

После чего на рабочем столе будет открыта новая вкладка, которая будет содержать результат сравнения двух графов (см. Рис. 19:).

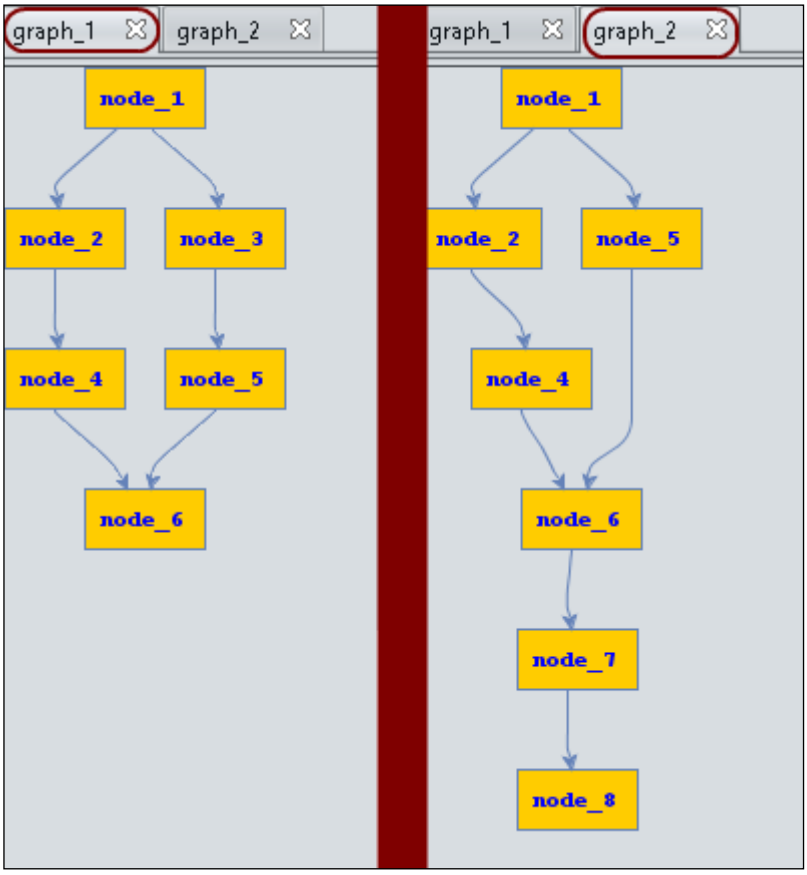

Рис. 18: Пример исходных графов

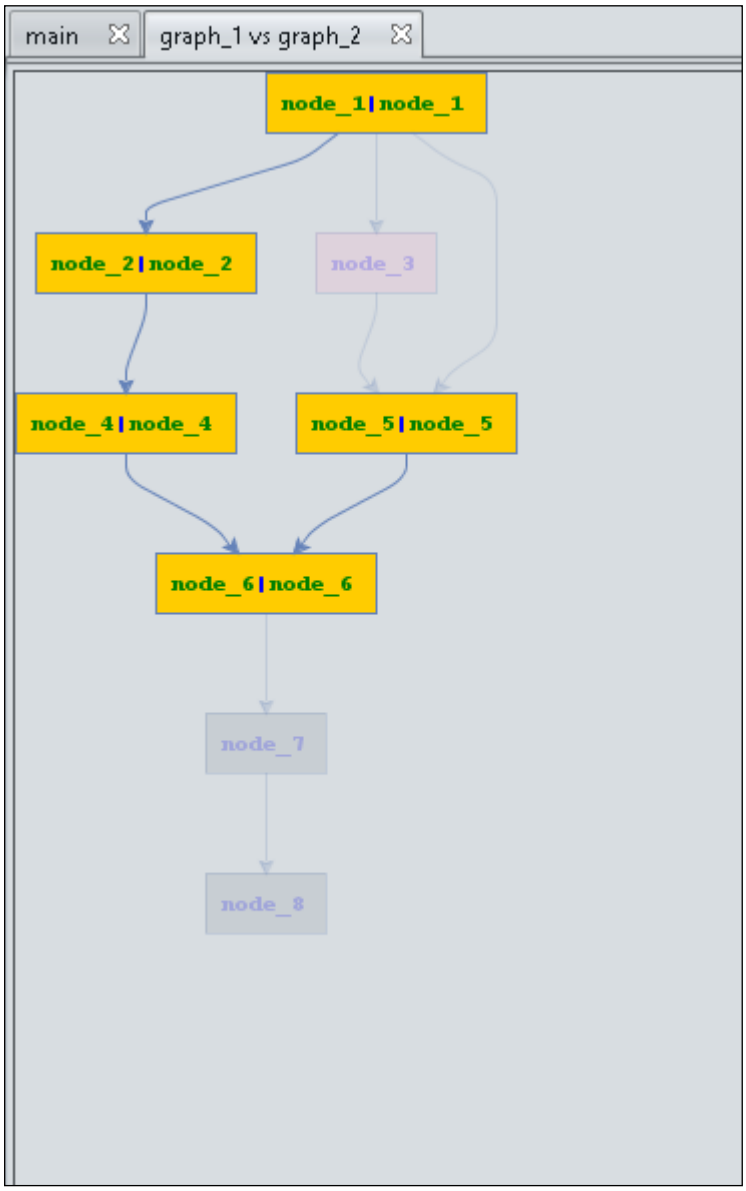

Рис. 19: Вкладка с результатом сравнения двух графов

На Рис. 19: выделен наибольший общий граф, и полупрозрачными цветами выделены элементы не вошедшие в него. Красным оттенком выделены не вошедшие вершины первого графа, а синим второго.

Пользователь имеет возможность изменить веса тех или иных атрибутов, тем самым повлиять на процесс сопоставления вершин.

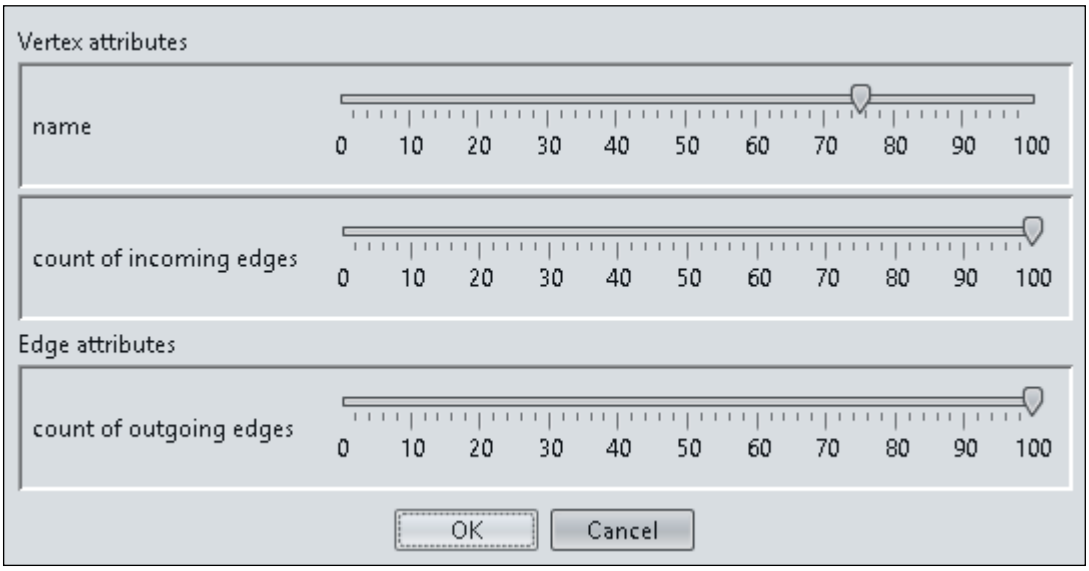

Рис. 20: Меню для изменения весов атрибутов

В меню на Рис. 21: пользователь имеет возможность выбрать строгое сравнение или мягкое:

1. При строгом сравнении значении двух атрибутов считаются одинаковыми только при их полном совпадении;
2. При мягком сравнении для сравнения числовых атрибутов используется следующая формула:

$$1 - \frac{|a-b|}{\max(a,b)}, \text{при условии, что } |a| + |b| > 0, a \geq 0, b \geq 0$$

Отрицательные a и b не сложно привести к нужному виду.

Для сравнения строковых атрибутов используется следующая формула:

$$\frac{(2 * i)}{length1 + length2}$$

Где i – позиция, в которой строки не совпадают или длина строки в случае если строки равны. length1 и length2 – длина первой и второй строки соответственно.

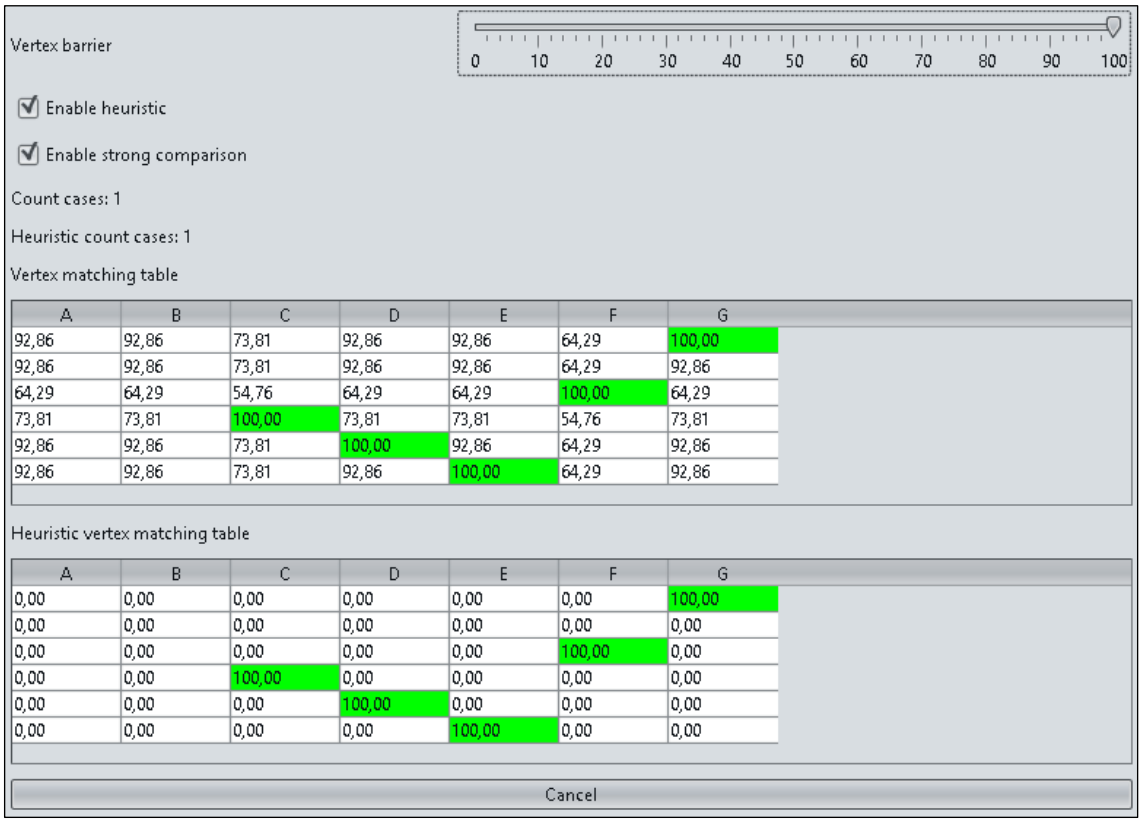

Рис. 21: Меню типа сравнения, процентного барьера при котором две вершины считаются одинаковыми, а так же включение и отключение эвристик

Так же в данном меню пользователь может оценить максимальное количество шагов перебора, которое потребуется сделать алгоритму для определения наибольшего общего подграфа с включенной эвристикой и без нее. В текущей реализации перед началом перебора, с включенной эвристикой, используется венгерский алгоритм. Это дает существенный прирост, когда пользователь выставляет значение барьера, при котором две вершины считаются одинаковыми ниже 100%.

Так же стоит отметить, что в любой момент поиска решения, пользователь может остановить этот процесс и получить какое-то решение, не обязательно наилучшее.

В результате эксплуатации данного инструмента было установлено, что алгоритм, предложенный в разделе 3.2.2.1, достаточно хорошо справляется с графами, элементы, которых содержат достаточно разнородные значения одних и тех же атрибутов и плохо применим в противном случае. Количество вершин в тестовых графах варьировалось от 10 до 1000.

3.2.2.8 Блокнот

Блокнот - инструмент, позволяющий загружать файлы с дополнительной информацией и связывать их с графовыми моделями. После чего пользователь может

перейти из элемента графовой модели к дополнительной текстовой информации. Например, это можно использовать для связи графа с исходным кодом.

Функциональность:

- позволяет загружать вспомогательную информацию для Графовых моделей, а так же связывать ее с ними (см. Рис. 22: номер 1 для открытия файла и номер 2 для связывания);
- позволяет просматривать текстовые файлы. Но без возможности редактирования содержимого;
- позволяет осуществлять поиск по тексту и выдавать в удобной для пользователя форме его результаты. Для этого есть механизм подсветки результатов поиска, а так же отображение результатов поиска на мини карте справа (см. Рис. 22:);
- позволяет работать одновременно с несколькими файлами.

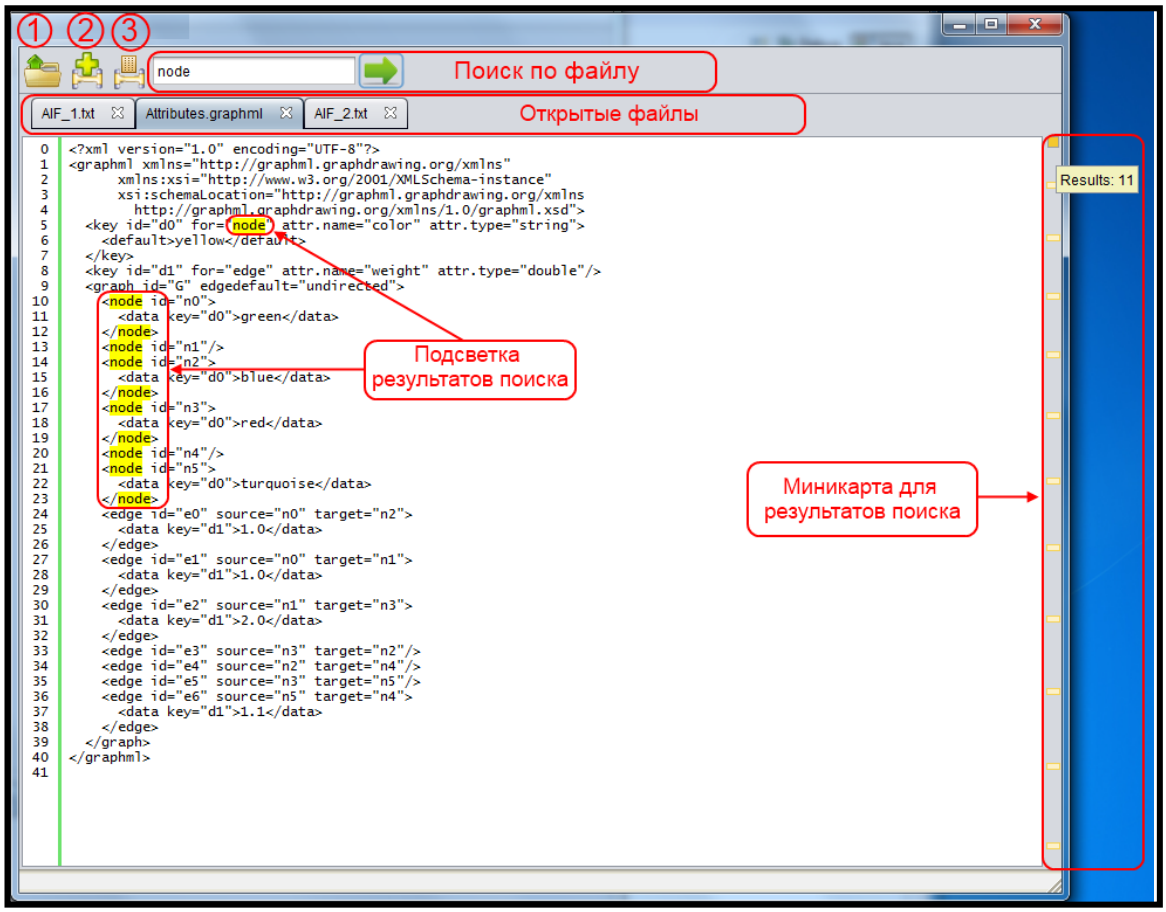

Рис. 22: Блокнот

Рассмотрим подробнее механизм связывания Графовой модели и файла содержащего вспомогательную информацию (см. Рис. 22: под номером 2, Рис. 23:). Файл со вспомогательной информацией должен содержать тэги вида:

- <anchor f1 = link_1/>some string – данный тэг будет относиться к одной строке,

идущей после него;

- <anchor f2 = link_1> some text</anchor> - данный тэг будет относиться ко всему тексту, заключенному в нем.

Стоит отметить, что эти тэги служат только для определения точки в тексте, в которую нужно перейти из Графовой модели. И в тексте они не отображаются.

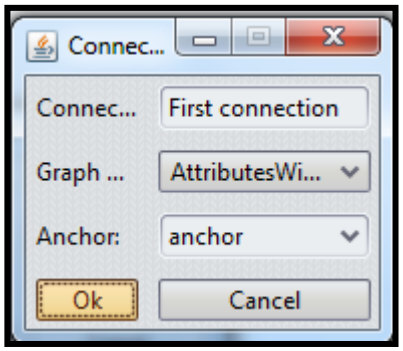

Рис. 23: Создание новой связи

Так же пользователю доступен список всех созданных им связей, между Графовыми моделями и файлами, содержащими вспомогательную информацию (см. Рис. 22: под номером 3, Рис. 24:). При желании пользователь может отредактировать существующие связи и удалить ненужные.

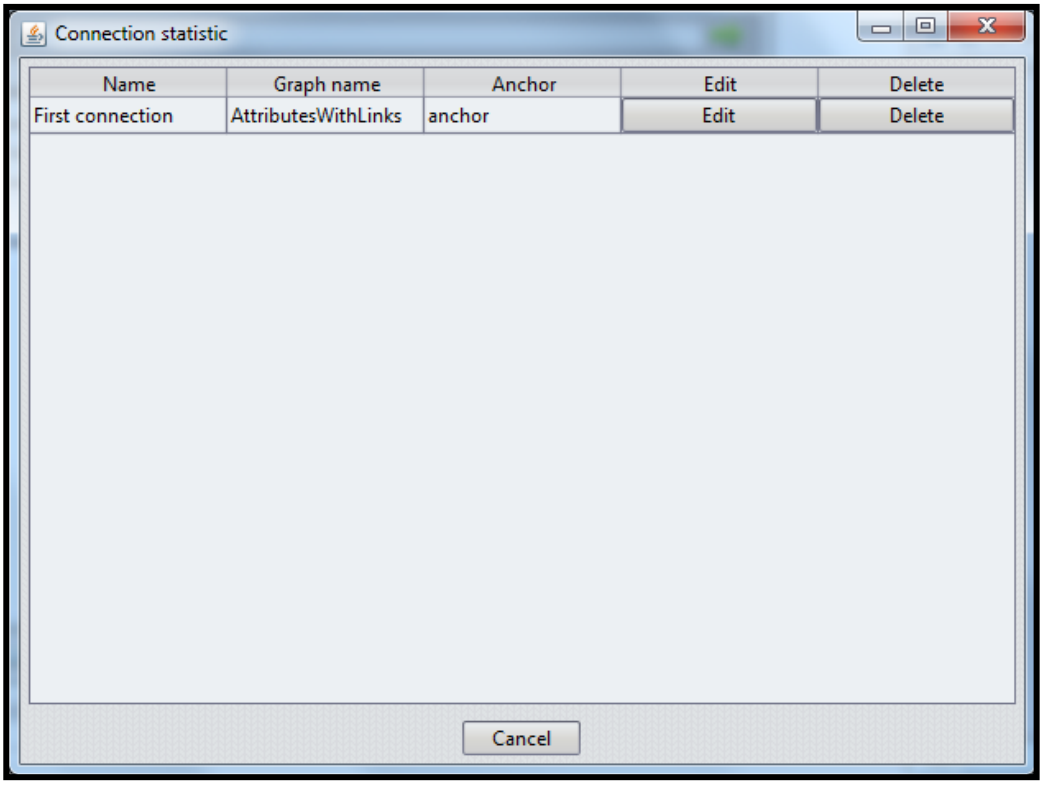

Рис. 24: Редактирование существующих связей

ГЛАВА 4 ТЕСТИРОВАНИЕ И СОПРОВОЖДЕНИЕ ПРОГРАММЫ

Для модульного тестирования программы был использован JUnit[14]. С его помощью было написано множество тестовых наборов, которые выполняют тестирование самых разных функциональных аспектов системы Visual Graph.

Для тестирования пользовательского интерфейса был составлен документ для тестирования. В данном документе описаны наборы тестов, которые инженер по тестированию должен сделать, чтобы убедиться, что Программа работоспособна.

Для облегчения сборки Программы был написан скрипт для Maven фреймворка, который автоматизирует процесс сборки проектов [13]. Данный скрипт собирает Программу из исходного кода и дополнительных ресурсов, таких как подключаемые библиотеки, картинки, инструкция для пользователей и т.п. Весь процесс сборки задокументирован в руководстве для пользователя и разработчика, которое распространяется вместе с Программой.

Не секрет, что иногда в программе происходят ошибки. Некоторые программы ограничиваются тем, что выводят сообщение об ошибке. А некоторые просто не сообщают об этом, и пользователь получает неправильный результат, но думает, что он правильный. В Программе, помимо сообщений об ошибке предусмотрено еще и логирование. Т.е. Программа ведет журнал, в который она записывает ход своей работы. При ошибке Программы, пользователь может выслать файл лога разработчикам. И у тех в свою очередь будет намного больше информации, для устранения ошибки, по сравнению с тем, если бы пользователь просто сказал, что у него произошла какая-то ошибка.

Программа сопровождается руководством для пользователя и разработчика. В нем описаны сборка программы из исходных кодов, запуск программы, а так же описаны средства, доступные пользователю для работы.

ЗАКЛЮЧЕНИЕ

За два года была проделана следующая работа:

1. изучение предметной области,
2. сбор системных требований к системе,
3. реинжиниринг архитектуры системы,
4. реинжиниринг существующих инструментов навигации по графам:
    a. миникарта,
    b. навигатор,
    c. атрибутная панель,
    d. фильтр,
    e. поисковая панель,
    f. блокнот.
5. разработка и реализация алгоритма позволяющего проводить сравнительный анализ двух графов и выдавать максимальный общий подграф этих графов,
6. разработка и реализация библиотеки, позволяющей сохранять в нужном формате промежуточное представление транслируемой программы в компиляторе Заказчика,
7. добавление поддержки dot (graphviz) формата.

Была написана статья “Визуализация графов при помощи программного средства Visual Graph” для сборника “Информатика в науке и образовании” [19].

Было принято участие в следующих научных конференциях:

1. 50-ая юбилейная международная научная студенческая конференция «Студент и научно-технический прогресс»[20].
2. Научно студенческая конференция лаборатории НГУ-Интел «Технологии высокопроизводительных вычислений» [21].
3. Всероссийская научно-практическая конференция «Наука. Технологии. Инновации» для студентов, аспирантов и молодых ученых, где данный проект получил диплом III степени [22].
4. Юбилейная II Международная Интернет - конференция молодых ученых, аспирантов и студентов «Инновационные технологии: теория, инструменты,

практика» (InnoTech 2010) [23].

Был получен документ от Заказчика о внедрении им системы в тестовую эксплуатацию.

План продолжения работы:

1. совершенствование существующих средств навигации,
2. поиск новых клиентов для Программы.

ПЕРЕЧЕНЬ УСЛОВНЫХ ОБОЗНАЧЕНИЙ

- **Visual Graph** – название системы (далее Программа или проект Visual Graph);
- **Входной язык** – язык, позволяющий описывать следующие сущности:
  - вершины и дуги графа;
  - графы с любым количеством вершин, дуг, вложенных графов;
  - произвольное количество атрибутов для любого графа, вершины или дуги.

  По умолчание, Входным языком является GraphML[4].
- **Входной граф** – файл, содержащий Графовую модель, описанный с помощью Входного языка;
- **Плагин** - программный модуль, подключаемый к Программе, предназначенный для расширения её возможностей;
- **Заказчик** - лицо (физическое или юридическое), заинтересованное в выполнении исполнителем работ;
- **ТЗ** - техническое задание;
- **Фреймворк** - программное обеспечение, облегчающее разработку и объединение разных компонентов большого программного проекта.

ЛИТЕРАТУРА

1. Касьянов, В.Н. Иерархические графы и графовые модели: вопросы визуальной обработки // Проблемы систем информатики и программирования. - Новосибирск, 1999. - С. 7-32.

2. Касьянов, В.Н., Евстигнеев, В.А. Графы в программировании: обработка, визуализация и применение. – Санкт-Петербург, 2003.

3. Гамма, Э., Хелм, Р., Джонсон, Р., Влисседс, Дж. Приемы объектно-ориентированного проектирования. Паттерны проектирования. – Санкт-Петербург, 2001.

4. Kuhn, Harold William. The Hungarian Method for the assignment problem // Naval Research Logistics Quarterly. 1955. - С. 83—97.

5. Munkres, James. Algorithms for the Assignment and Transportation Problems // Journal of the Society for Industrial and Applied Mathematics. 1957 March. - С. 32—38.

6. Домашняя страница проекта Java. URL: http://www.java.com (дата обращения: 10.05.2013)

7. Домашняя страница проекта SQLite. URL: http://www.sqlite.org (дата обращения: 10.05.2013)

8. Домашняя страница проекта sqlite4java. URL: http://code.google.com/p/sqlite4java/ (дата обращения: 10.05.2013)

9. Домашняя страница проекта JGraphX. URL: http://www.jgraph.com (дата обращения: 10.05.2013)

10. Спецификация GraphML формата. URL: http://graphml.graphdrawing.org/specification.html (дата обращения: 10.05.2013)

11. Спецификация DOT формата (Graphviz формата). URL: http://graphml.graphdrawing.org/specification.html (дата обращения: 22.04.2013)

12. Домашняя страница проекта Substance. URL: http://insubstantial.github.io/insubstantial/substance/ (дата обращения: 10.05.2013)

13. Домашняя страница проекта Maven. URL: http://maven.apache.org/ (дата обращения: 05.05.2013)

14. Домашняя страница проекта JUnit Home URL: http://www.junit.org (дата обращения: 10.05.2013)

15. Проприетарный аналог aiSee. URL: http://www.aisee.com (дата обращения:

15.05.2011)

16. Проприетарный аналог yEd. URL: http://www.yworks.com (дата обращения: 15.05.2011)

17. Аналог Cytoscape. URL: http://www.cytoscape.org/ (дата обращения: 15.05.2011)

18. Домашняя страница проекта Visual Graph. URL: https://code.google.com/p/visualgraph (дата обращения: 28.04.2013)

19. Золотухин, Т.А. Визуализация графов при помощи программного средства Visual Graph // Информатика в науке и образовании. - Новосибирск, 2012. - С. 135-148.

20. Золотухин, Т.А., Колбин, Д.С. Visual Graph: интерактивная система визуализации графовых моделей // Информационные технологии // материалы 50-ой юбилейной международной научной студенческой конференции «Студент и научно-технический прогресс». – Новосибирск, 2012. – С. 11.

21. Золотухин, Т. А., Колбин, Д. С. Универсальная интерактивная среда визуализации атрибутированных иерархических графовых моделей // Технологии высокопроизводительных вычислений: материалы научной студенческой конференции Лаборатории НГУ-Интел. — Новосибирск, 2010. - С. 60–66.

22. Золотухин, Т. А., Колбин, Д. С. Универсальная интерактивная среда Visual Graph для визуализации атрибутированных иерархических графовых моделей // Наука. Технологии. Инновации // Материалы всероссийской научной конференции молодых ученых в 4-х частях. — Новосибирск, 2010. - С. 17–18.

23. Золотухин, Т.А., Колбин, Д.С. Универсальная интерактивная среда визуализации атрибутированных иерархических графовых моделей // Юбилейная II Международная Интернет - конференция молодых ученых, аспирантов и студентов «Инновационные технологии: теория, инструменты, практика» (InnoTech 2010) // (http://www.conference.msa.pstu.ru/) (дата обращения: 10.05.2013)