Т.А. Золотухин

ВИЗУАЛИЗАЦИЯ ГРАФОВ ПРИ ПОМОЩИ ПРОГРАММНОГО СРЕДСТВА VISUAL GRAPH[1]

ВВЕДЕНИЕ

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

Многие структуры данных, представляющие практический интерес в математике и информатике, могут быть представлены в виде графов. Одним из основных классов является класс иерархических и/или атрибутированные графов[1].

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

В данной статье будет рассмотрено одно из таких программных средств, которое предназначено для визуализации и поиска информации в иерархических атрибутированных графах (далее – графовая модель). Данное программное средство называется Visual Graph и разрабатывается при содействии лаборатории НГУ-Интел в рамках проекта по созданию системы для визуализации графовых моделей[2].

Также стоит отметить, что Visual Graph использует лицензию BSD, что позволяет применять возможности данного проекта практически в любых целях.

[1] Работа выполнена при частичной финансовой поддержке Российского фонда фундаментальных исследований (грант РФФИ № 12-07-00091)

ОБЛАСТЬ ПРИМЕНЕНИЯ

В ходе развития компиляторов было создано несколько структур, которые представляются в виде графов: синтаксические деревья, графы потоков управления и графы вызовов. Каждый из этих типов графов имеет свое практическое применение. Так, синтаксические деревья используются при построении внутреннего представления программы в компиляторах или интерпретаторах. Графы потока управления – для оптимизации в компиляторах и утилитах статического анализа кода, а графы вызовов – при отладке программ. Все эти графы объединяет то, что их можно рассматривать как иерархические и атрибутированные 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 решения данной проблемы разработчики выбрали способ, в котором либо сам компилятор, либо вспомогательная программа переводит граф из внутреннего представления в файл одного из поддерживаемых программным средством Visual Graph форматов.

После описанной выше процедуры сохранения программное средство Visual Graph сможет прочитать эту графовую модель из файла, визуализировать ее и предоставить пользователю средства навигации по ней.

СУЩЕСТВУЮЩИЕ ПРОБЛЕМЫ И ИХ РЕШЕНИЯ

При разработке программного средства Visual Graph разработчики ь столкнуться с массой проблем. В данном разделе будут изложены основные из них, а так же то, как они были решены в рамках программного средства Visual Graph.

Начать стоит с проблемы хранения графов вне программного средства Visual Graph, т.е. представление графов в файловой системе, в текстовом виде. Для представления графов в текстовом виде существует множество языков описания обычных графов, но для иерархических атрибутированных графов количество таких языков невелико. Одним из таких языков является GraphML[3].

GraphML – это язык описания графов, основанный на XML. Этот формат позволяет описывать направленные, ненаправленные и смешанные графы, гиперграфы и иерархические графы, специфичные для приложений

атрибуты. Соответственно, язык GraphML полностью поддерживает иерархические атрибутированные графы.

Другой проблемой является хранение графов внутри программного средства Visual Graph. Размер входного графа может достигать сотен 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, было принято решение об использовании реляционной базы данных.

В качестве реляционной базы данных была выбрана встраиваемая база данных SQLite[4]. В отличие от большинства популярных реляционных баз данных, для SQLite не требуется установка сервера, а вся клиент-серверная архитектура сводится к работе с файлами.

Следующей проблемой, с которой пришлось столкнуться при разработке, стала проблема расширяемости системы. Данная проблема возникает из-за того, что такие части программного средства, как навигация, визуализация и анализ графов, представляются набором средств, который может расширяться как разработчиками самого программного средства Visual Graph, так и сторонними разработчиками.

Для построения расширяемой системы было принято решение о добавлении поддержки плагинов. На данный момент вся функциональность программного средства Visual Graph является набором плагинов, которые могут взаимодействовать друг с другом. Взаимодействие происходит за счет системы уведомлений и запросов. Таким образом, каждый плагин может отправить системе запрос или уведомление, и та, в свою очередь, разошлет его по всем плагинам. Плагин, исполняющий чей-либо запрос, также 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.

Следующей проблемой является проблема навигации по графовым моделям. Как известно, нет какого-то универсального набора средств навигации, который бы удовлетворял любым потребностям любого пользователя для решения любой его задачи. Поэтому было принято решение ориентироваться на область применения, описанную выше, и разработать для нее следующие инструменты:

1. Рабочий стол и миникарта (рис. 1). Рабочий стол состоит из набора вкладок, которые пользователь открывает для визуализации выбран-

ной части графовой модели. Каждая вкладка состоит из рабочей области и нерабочей вкладки, соответственно. 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. Нерабочая область – область, которую пользователь не видит, но может туда попасть, используя скроллинг. Миникарта позволяет увидеть весь граф целиком в текущей вкладке, а также текущую рабочую область.

2. Навигатор, визуализирующий графовые модели, с которыми в данный момент работает пользователь в виде дерева (рис. 1). В данном дереве отображены только вершины, т.к. отображение ребер не будет нести никакой смысловой информации, но будет очень сильно засорять рисунок. Для быстрого поиска по дереву реализована строка быстрого 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.
3. Атрибутная панель, которая 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, после чего отметить в атрибутной панели галочками те атрибуты, которые он хочет визуализировать (рис.10).
4. Фильтр – инструмент, который использует тот факт, что пользователь работает с атрибутированными графами (рис.2). Данный инструмент осуществляет поиск вершин и ребер по заданным условиям в текущей вкладке. Построение условий осуществляется за счет атрибутов и их значений. На рис.2 показан механизм задания выражения, состоящего из логических связок, скобок и условий вида <имя атрибута = его значение (возможно подстроки)>. После того как пользователь закончит формирование выражения, оно исполнится на элементах (вершины и ребра) графовой модели, находящихся в текущей вкладке.

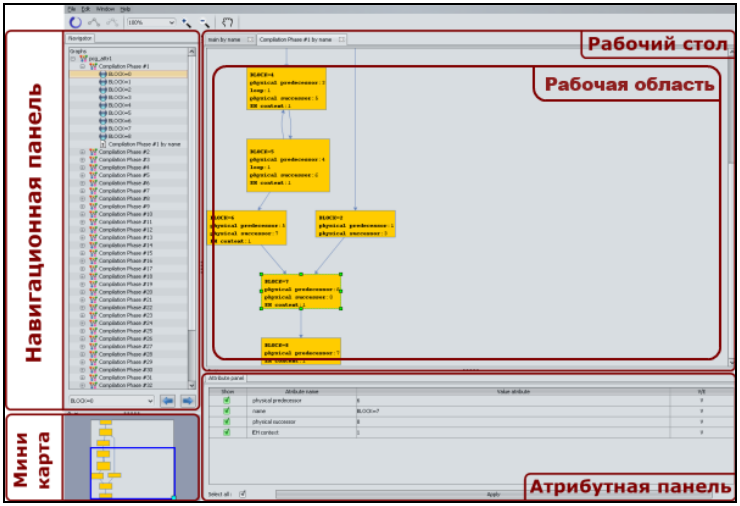

Рис. 1. Пользовательский интерфейс программы Visual Graph

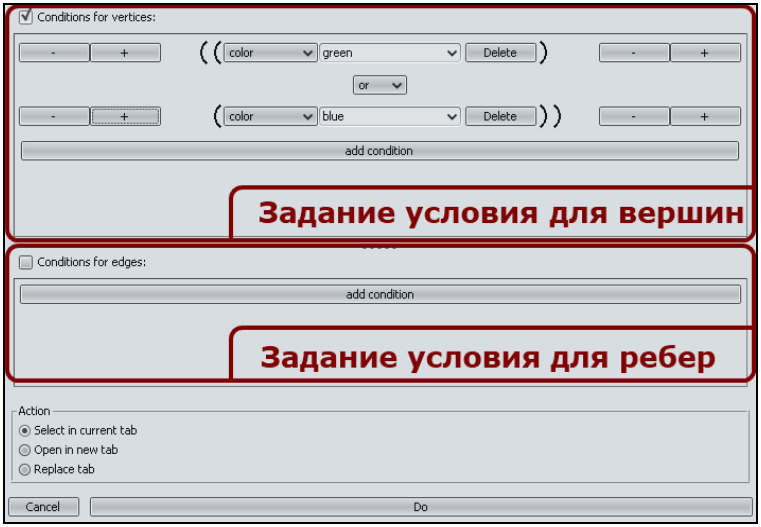

Рис. 2. Фильтр

В результате пользователь получит набор вершин и ребер, которые будут удовлетворять заданным им условиям (рис. 3).

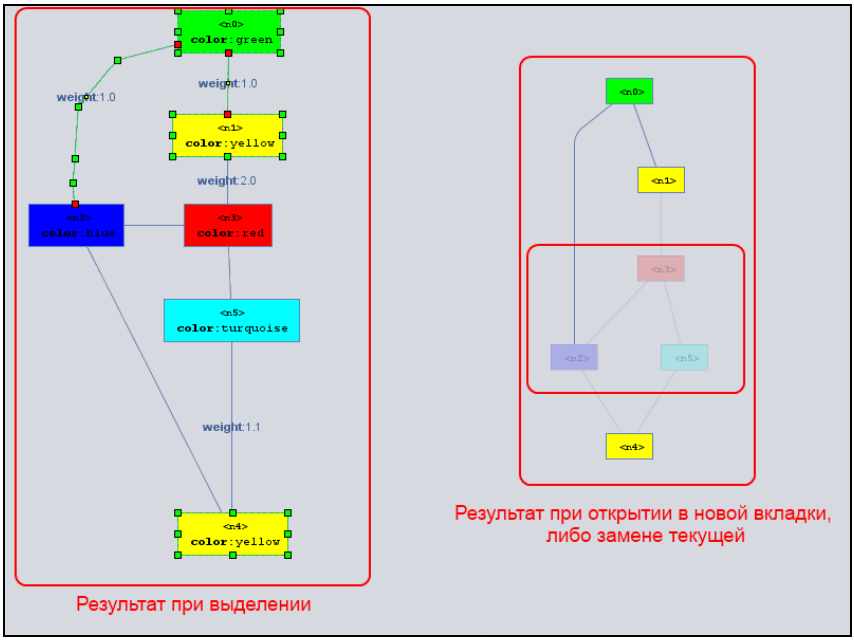

Рис. 3. Результат работы фильтра

5. Поисковая панель – инструмент, который, как и фильтр, использует тот факт, что пользователь работает с атрибутированными графами. В отличие 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. Такой поиск даёт т.н. результирующее дерево (0), в котором красными крестиками отмечены вершины, не удовлетворяющие заданному условию.

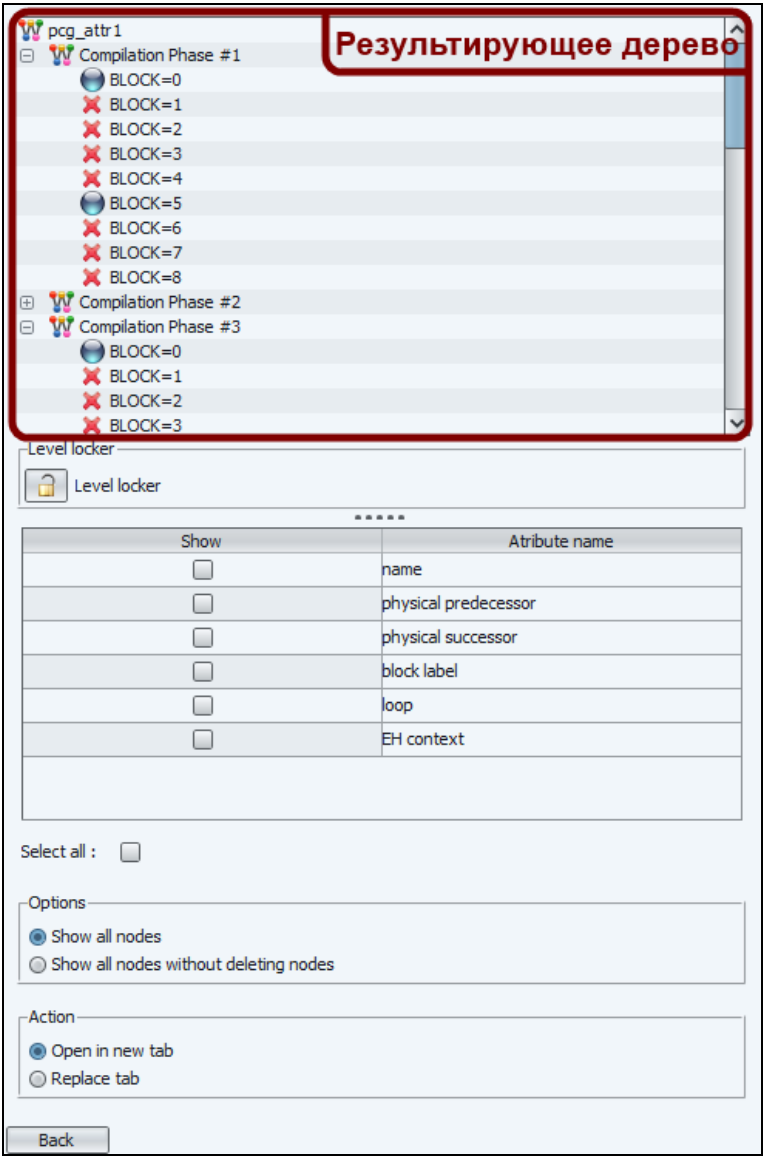

Рис. 4. Результат работы поисковой панели

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.

6. Блокнот – инструмент, позволяющий загружать файлы с дополнительной информацией и связывать их с графовыми моделями. После этого пользователь может перейти из элемента графовой модели к дополнительной текстовой информации. Например, это можно использовать для связи графовой модели с исходным кодом.

К средствам навигации также можно отнести и средства структурного анализа, т.е. различные алгоритмы, применяемые к графам и помогающие пользователю получить различного рода информацию: подсветку кратчайшего пути в графе, подсветку циклов и т.п.

В программе Visual Graph к средствам структурного анализа можно отнести подсветку кратчайшего пути, подсветку циклов, а также сравнение графов с подсветкой различий в них.

Отдельно хотелось бы остановиться на инструменте, позволяющем проводить сравнительный анализ двух графов и показывать наибольший общий подграф, а в случае, если графы изоморфны – сообщать об этом. Данного инструмента нет ни в одной из трех рассмотренных ниже программ-аналогов, и, в отличие от инструментов поиска пути и циклов, он наиболее сложен в реализации.

Как известно, алгоритм поиска изоморфизма двух графов принадлежит классу NP и в основном решается полным перебором с применением эвристик для уменьшения числа вариантов для перебора. Алгоритм, разрабатываемый для данного инструмента, 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). При этом нуль означает кардинальное несовпадение вершин, а единица, наоборот, полное их совпадение
3. К полученному двудольному графу применяется венгерский алгоритм решения задачи о назначениях для получения максимального паросочетания максимального веса [8, 9]. Очевидно, что такое па-

росочетание может быть не в единственном числе и далее остается перебрать все эти варианты.

Несмотря на то, что в конце алгоритм сводится 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 программы Visual Graph, 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.

Последней проблемой, которая будет рассмотрена в данном разделе, является проблема отображения графов, а именно раскладка графа на плоскости.

Задача раскладки не имеет и не может иметь оптимального решения в связи с тем, что наборы критериев для оценки качества раскладки различны для различных графов. Для некоторых графов одни критерии будут вносить большую значимость в читаемость графа, чем другие. Тем не менее, можно подобрать виды раскладок, их примерные параметры и алгоритмы для конкретных классов графов, которые обычно визуализируются.

Рассмотрим графы, которые обычно генерируются при работе компилятора:

1. Графы управления. Они обычно имеют относительно небольшое количество ребер и структуру, близкую к иерархической. Как правило, поуровневая раскладка хорошо справляется с такими видами графов. В зависимости от объема графа можно использовать эвристики или алгоритмы для минимизации ширины или высоты получаемой раскладки, а также пересечений между ребрами.
2. Синтаксические деревья. Существует множество техник для отображения деревьев. Можно воспользоваться алгоритмами для упорядочивания узлов или алгоритмами радиальной раскладки дерева.

ОСОБЕННОСТИ ПРОГРАММНОГО СРЕДСТВА VISUAL GRAPH И ЕГО АНАЛОГОВ

В настоящее время на рынке представлен достаточно широкий круг программных средств для работы с графами. Наиболее известными являются aiSee[5], yEd[6] и cytoscape[7].

Область применения этих программных средств частично совпадает с областью применения программного средства Visual Graph. Следует пояснить слово «частично». В данном контексте оно означает, что их область применения шире, т.к. многие из них рассчитаны на редактирование существующих графовых моделей и создание новых. Программное средство Visual Graph ориентировано исключительно на просмотр существующих графовых моделей без возможности их редактирования.

Основные задачи, решаемые подобными программными средствами – это отображение графов и навигация по ним. Поэтому в рамках этих двух задач и будут рассмотрены особенности каждого из вышеперечисленных средств.

Вначале стоит сказать несколько слов о каждом из рассматриваемых нами программных средств.

aiSee – программное средство, которое автоматически рисует раскладку графа, описанного на языке GDL. Затем пользователь может интерактивно исследовать, данный граф, отпечатать его и сохранить в различных форматах. 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, таких как 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 – свободное программное обеспечение с открытым исходным кодом для визуализации и анализа сетей. Основная область применения данной программы – это биоинформатика. Данное программное средство стало очень популярным благодаря тому, что оно легко расширяется, позволяя сторонним разработчикам писать для него различные плагины.

yEd представляет собой программное средство, которое может быть использовано для быстрого создания и редактирования высококачественных диаграмм. Создание диаграмм происходит вручную или с помощью импортирования из внешних данных. После этого к полученной диаграмме можно применить богатый набор алгоритмов для проведения анализа с последующим получением необходимой информации.

Каждое из рассматриваемых программных средств 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 отметить присутствие алгоритмов из yFiles, которые используются в yEd. Стоит также отметить, что результаты, получаемые в yEd, намного лучше, чем аналогичные результаты в Cytoscape с настройками раскладчиков по умолчанию.

Программное средство Visual Graph предоставляет несколько раскладчиков, главным из которых является модифицированный иерархический раскладчик, максимально адаптированный 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 программа aiSee для решения данной задачи:

1. Рабочий стол, который визуализирует графовую модель целиком, выдавая пользователю статичную картинку, которую нельзя изменять, например, передвигая элементы.
2. Поисковый 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 программа yEd для решения данной задачи:

1. Рабочий стол схожий с рабочим столом, который предоставляет aiSee. Но, в отличие от aiSee:
   a. есть возможность работать с несколькими графовыми моделями одновременно в одном экземпляре программы (создается вкладка для каждой графовой модели),
   b. есть возможность редактирования элементов графовой модели, начиная от смены их положения и размеров до задания атрибутов, влияющих на их визуализацию
2. Навигатор, отображающий графовую модель в виде дерева. Стоит отметить, что данный инструмент отображает дерево только для

той графовой модели, с которой в данный момент работает пользователь.

3. Миникарта, показывающая всю графовую модель целиком.

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

1. Рабочий стол, наподобие того, что предоставляет yEd. Отличие заключается в том, что программа Cytoscape не умеет работать с иерархическими графами.
2. Миникарта, наподобие миникарты в программном средстве yEd.
3. Атрибутная панель, которая позволяет отображать текущие атрибуты и задавать новые. Данный инструмент выглядит как таблица, по горизонтали которой расположены имена атрибутов, по вертикали – список выделенных вершин, а в ячейках находятся соответствующие значения того или иного атрибута для той или иной вершины.
4. Фильтр, позволяющий осуществлять поиск вершин и ребер по заданным условиям на атрибутах.

Выделим основные средства навигации, которые предоставляет программное средство Visual Graph (подробное описание всех нижеперечисленных инструментов можно найти в разделе “ Существующие проблемы и их решения”):

1. Рабочий стол, визуализирующий части графовой модели. К особенностям данного инструмента можно отнести:
   1) возможность влиять на визуализацию элементов посредством задания определенных атрибутов;
   2) возможность работать с несколькими графовыми моделями одновременно, в одном экземпляре программы;
   3) возможность редактирования элементов графовой модели (изменения положения элементов и их размеров);
   4) возможность просмотра интересующей пользователя области графовой модели в другом окне или вкладке. Это одна из важнейших особенностей данного инструмента. Человеку для понимания чего-то сложного (а графовые модели, безусловно, сложны) нужно сложное разделить на множество маленьких простых частей, которые он сможет легко проанализировать;
   5) возможность визуализации пользовательских атрибутов.
2. Навигатор, наподобие навигатора в программном средстве yEd. Основным отличием является то, что в yEd навигатор отображает графовую модель, находящуюся в текущей вкладке, а не все графо-

вые модели, с которыми работает пользователь, как это сделано в Visual Graph. Несложно понять, что в случае с yEd во главе угла стоит рабочий стол, тогда как в Visual Graph главным элементом является навигатор.

3. Миникарта, наподобие миникарты в программном средстве yEd.
4. Атрибутная панель, отображающая набор атрибутов у выделенных элементов графовой модели, а также позволяющая управлять их визуализацией на рабочем столе.
5. Фильтр и поисковая панель осуществляют поиск элементов по заданному выражению. К особенностям данного инструмента можно отнести:
   1) возможность задания сложных булевских выражений, содержащих скобки и разнотипные знаки между ними;
   2) использование типов атрибутов при задании выражения.
6. Блокнот. Несмотря на то, что данный инструмент не обладает большинством стандартных функций, таких как подсветка синтаксиса или поиск с использованием регулярных выражений, его функциональности вполне хватает для решения тех задач, которые встают перед пользователем программного средства Visual Graph. Функциональность включает:
   1) возможность открывать множество текстовых файлов и связать их с графовыми моделями для последующего перехода из элемента графовой модели к фрагменту в текстовом файле;
   2) подсветку результатов поиска.
7. Средства структурного анализа, такие как поиск циклов и кратчайших путей, а также сравнительный анализ двух графов. Сложно охарактеризовать плюсы и минусы данных инструментов, но можно с уверенностью сказать, обойтись без них практически невозможно: стоит лишь представить, сколько времени уйдет на то, чтобы вручную сравнить два графа со 100 вершинами, не говоря уже о более крупных графах.

ЗАКЛЮЧЕНИЕ

В данной статье были рассмотрены проблемы, возникшие в ходе разработки программного средства Visual Graph, его область применения, а также было произведено исследование схожих программных средств. Результаты исследования показали, что программное средство Visual Graph обла-

дает множеством достоинств и не уступает аналогам в области визуализации графов, используемых в компиляторах.

Также стоит отметить, что использование программного средства Visual Graph не ограничено только визуализацией графов, применяемых в компиляторах. Данное программное средство может быть использовано и в других смежных областях, в которых необходима визуализация графов и навигация по ним.

СПИСОК ЛИТЕРАТУРЫ

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. – 1104 с.
3. Спецификация GraphML. URL: http://graphml.graphdrawing.org/specification.html (дата обращения: 15.05.2011)
4. Домашняя страница проекта SQLite. URL: http://www.sqlite.org (дата обращения: 15.05.2011)
5. Проприетарный аналог aiSee. URL: http://www.aisee.com (дата обращения: 15.05.2011)
6. Проприетарный аналог yEd. URL: http://www.yworks.com (дата обращения: 15.05.2011)
7. Cytoscape. URL: http://www.cytoscape.org/ (дата обращения: 15.05.2011)
8. Kuhn H. W. The Hungarian Method for the assignment problem. Naval Research Logistics Quarterly, 2:83—97, 1955. Kuhn's original publication.
9. Munkres J. Algorithms for the Assignment and Transportation Problems, Journal of the Society for Industrial and Applied Mathematics, 5(1):32—38, 1957 March.