3. Для сетевых задач, работающих в режиме реального времени, протокол AODV является более предпочтительным, чем протоколы DSR и DSDV, т.к. при увеличении числа узлов время доставки пакетов в сети при использовании протокола AODV мало изменяется, а указанные сетевые задачи чувствительны к флуктуациям времени доставки пакетов.

4. Для сетей с малым количеством узлов и низкой мобильностью протокол DSDV более эффективен, чем протоколы DSR и AODV.

Таким образом, можно сделать следующие выводы:

1. Если преобладающим классом сетевых задач в узлах сети является третий или четвертый (см. Таблицу №1), то при любом числе узлов и мобильности предпочтение необходимо отдать протоколу AODV.

2. Если номер преобладающего класса сетевых задач в узлах сети не третий и четвертый, а число узлов и средняя мобильность малы, то предпочтительным является протокол DSDV.

3. Если номер преобладающего класса сетевых задач в узлах сети не третий и не четвертый, а число узлов или средняя мобильность высоки, то предпочтительным протоколом является DSR.

При этом под средней мобильностью понимается среднее арифметическое мобильности всех узлов. Номер преобладающего класса рассчитывается как среднее арифметическое номеров классов задач в узлах сети. Для различных типов запущенных сетевых приложений на узлах сети (например, Microsoft Otlook Express) известен номер класса задачи. При запуске нескольких сетевых задач на узле определяется среднее арифметическое номеров запущенных сетевых задач.

При выполнении условия пункта 1 предпочтительным протоколом является -AODV. В противном случае выбор протокола зависит от числа узлов, мобильности узлов, номера преобладающего класса. Правила, позволяющие точно выбрать наиболее предпочтительный протокол DSDV или DSR в зависимости от конкретных значений числа узлов, мобильности узлов, номера преобладающего класса как видно из пунктов 2 и 3 не определены. Известны отдельные примеры правильного выбора предпочтительного протокола, полученные экспериментальным путем для некоторых значений средней мобильности узлов, числа узлов и номера преобладающего класса. Следовательно, необходимо разработать формально-математически аппарат, позволяющий выбрать предпочтительный протокол для различных значений мобильности, числа узлов и номера преобладающего класса задач. В качестве такого аппарата обосновано применение нейронных сетей.

Таким образом, разработан эффективный инструментарий выбора предпочтительного протокола маршрутизаций AdHoc – сетей, основанный на использовании математического аппарата нейронных сетей. Показано, что применение разработанной инструментария может давать выигрыш в скорости доставки пакетов более чем в 2 раза.

Литература

1. Винокуров В.М, Пуговкин А.В. Маршрутизация в беспроводных AdHoc - сетях // Доклады ТУСУРа №2 (22), часть 1 – 2010, 288-292

2. Таненбаум Э. Компьютерные сети. 4-е изд. Спб.: Питер, 2003. – 992 с.

3. Sapna S. Kaushik, P.R.Deshmukh. Comparison of effectiveness of AODV, DSDV and DSR routing protocols in mobile ad hoc networks //International Journal of Information Technology and Knowledge Management, July-December 2009, Volume 2, No. 2, pp. 499-502.

4. Комашинский В.И., Смирнов Д.А. Нейронные сети и их применение в системах управления и связи. М.: Горячая линия Телеком, 2003. – 94 с.

5. Ясницкий Л.Н. Введение в искусственный интеллект. М.: Издательский центр «Академия», 2005. – 176 с.

6. Рогозин О.В. Комплексная оценка эффективности инновационного проекта на основе анализа качественных характеристик, журнал « Открытое образование»; №6, 2010, www.e-joe.ru.

РАЗРАБОТКА ГРАФИЧЕСКОГО РЕДАКТОРАДЛЯ ГЕОДЕЗИЧЕСКОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ

Автор: Зуев Дмитрий Геннадьевич, студент 6 курса**,** Питухин Пётр Васильевич, доцент
Руководитель: Питухин П. В.
Образовательное учреждение: Филиал «Протвино» университета «Дубна» Кафедра информационных технологий.

GRAPHIC EDITOR DEVELOPMENT FOR GEODESIC INFORMATION SYSTEM

В работе представлены результаты разработки программы графического редактора для построения продольных и поперечных профилей линейных объектов на основе топографической съёмки.

Геодезические информационные системы (ГИС) всегда были и остаются востребованными в производстве и в обществе. Важнейшим элементом ГИС является графическое представление. Именно поэтому задача по разработке графического редактора является до сих пор актуальной.

После проведения топографо-геодезических изысканий составляется ситуационный план трассы в масштабе 1:5000-1:2000. На основе этих планов выполняются продольные и поперечные профили линейных объектов в пикетных и плюсовых точках.

Для разработки и оформления графических материалов проектов на предприятии используется система автоматизированного проектирования и черчения «nanoCAD». Эта система может служить для архитектурного проектирования и черчения технологических объектов. С помощью графического пакета «nanoCAD» делаются продольные и поперечные профили линейных объектов.

Необходимо отметить, что построение профиля производится такими графическими примитивами как линии, прямоугольники и штриховки. При этом автоматизация производимых пользователем действий достигается только за счёт возможности копирования повторяющихся элементов и автоматической расстановки размерных линий.

Решением такого недостатка может являться разработка собственных встроенных программ для используемых программных комплексов, автоматизирующих работу пользователя. Однако практически все распространённые графические пакеты являются лицензионными программными продуктами с закрытым кодом, что, порой, не позволяет организовать необходимые расширения этих программ. Также велик риск отказа разработанных программ в случае перехода на новую версию программного продукта.

Целью данной работы явилась разработка узкоспециализированного программного продукта, позволяющего максимально упростить и автоматизировать создание про
дольных и поперечных профилей линейных объектов.

Разработка приложения осуществлялась в свободной интегрированной среде разработки модульных кроссплатформенных приложений Eclipse for RCP and RAP Developers, версии Juno SR 2 [1,2]. Также использовались такие расширения Eclipse, как Graphical Editing Framework (GEF) – структура программной системы, служащая для создания графических редакторов произвольного назначения и Eclipse Modeling Framework (EMF) – свободная структура программной системы, основанная на Eclipse и служащая для генерации кода, инструментов и прочих приложений на основе структурированной модели данных.[3,5] Для проектирования и реализации пользовательского интерфейса использовалась библиотека с открытым исходным кодом Standard Widget Toolkit (SWT), которая позволяет реализовать единый графический интерфейс для всех платформ, на которых доступно использование Java Virtual Machine (JVM). Реализация приложения осуществлялась на объектно-ориентированном языке программирования Java.[4]

В настоящее время на библиотеках GEF и EMF построена технология Eclipse Graphical Modelling Framework (GMF), предназначенная для быстрой разработки графических редакторов. Архитектура построенного с помощью GMF пакета основана на шаблоне проектирования Model/View/Controller (MVC), где для создания моделей используется EMF, а для создания представлений и контроллеров GEF.[3,5]

Процесс создания визуальных редакторов в GMF состоит из следующих шагов:

1. Разработка доменной модели. Доменная модель разрабатывалась с помощью графического редактора Ecore (входящего в состав GMF), импортирована из XMI (XML Metadata Interchange) документа, из набора аннотированных Java-интерфейсов или из XML-схемы.
2. Разработка модели графического представления – описание графической нотации разрабатываемого языка. Эта модель содержит информацию об элементах, которые создаются с помощью GEF.
3. Разработка модели инструментария. В ней описываются используемые в редакторе инструментальные средства: палитра объектов, меню графических объектов и т.п.
4. Разработка модели отображения – описание связки доменной модели и графических объектов, а также вспомогательных средств, описание ограничений на OCL или Java.
5. Создание модели генератора – промежуточного представления разрабатываемого редактора. Эта модель автоматически генерируется по модели соответствия и дополняется «вручную».
6. Генерация кода целевого графического редактора, запуск отладочного экземпляра Eclipse Application и отладка. Создаваемый модуль редактора встраивается в среду исполнения (runtime) GMF, которая в свою очередь осуществляет синхронизацию созданных моделей между собой в процессе работы пользователя с редактором.

Данная технология позволяет создавать независимые от Eclipse графические редакторы.[3]

Разработанная программа позволяет сократить время, затрачиваемое специалистом при построении профилей, за счёт использования и модификации библиотечных элементов, автоматизации расчётов и построения рельефа поверхности на соответствующем отрезке профиля с учётом горизонтального и вертикального масштабирования, что ведёт к значительному уменьшению количества выполняемых пользователем операций.

Рабочая область программы разделена на две части (см рис.1). В верхней части производятся действия по работе с топографической съёмкой. Пользователь наносит поверх топологической съёмки линии необходимых профилей. После чего, по ближайшим высотным отметкам, определяет области из замкнутых ломаных линий в местах построения профилей и задаёт этим областям соответствие библиотечным элементам. Каждый библиотечный элемент представляет собой группу графических примитивов и обладает определённым набором изменяемых параметров, таких как размер, структура, состав и т.д.

Нижняя часть рабочей области программы предназначена для работы с автоматически построенным профилем. Для этого, на основе введённых пользователем данных, производятся вычисления высотных отметок на

пересечении границ областей и линии профиля, определение глубин залегания инженерных коммуникаций, пересекающих линию профиля, и выполняется расчёт уклонов для каждого библиотечного элемента.

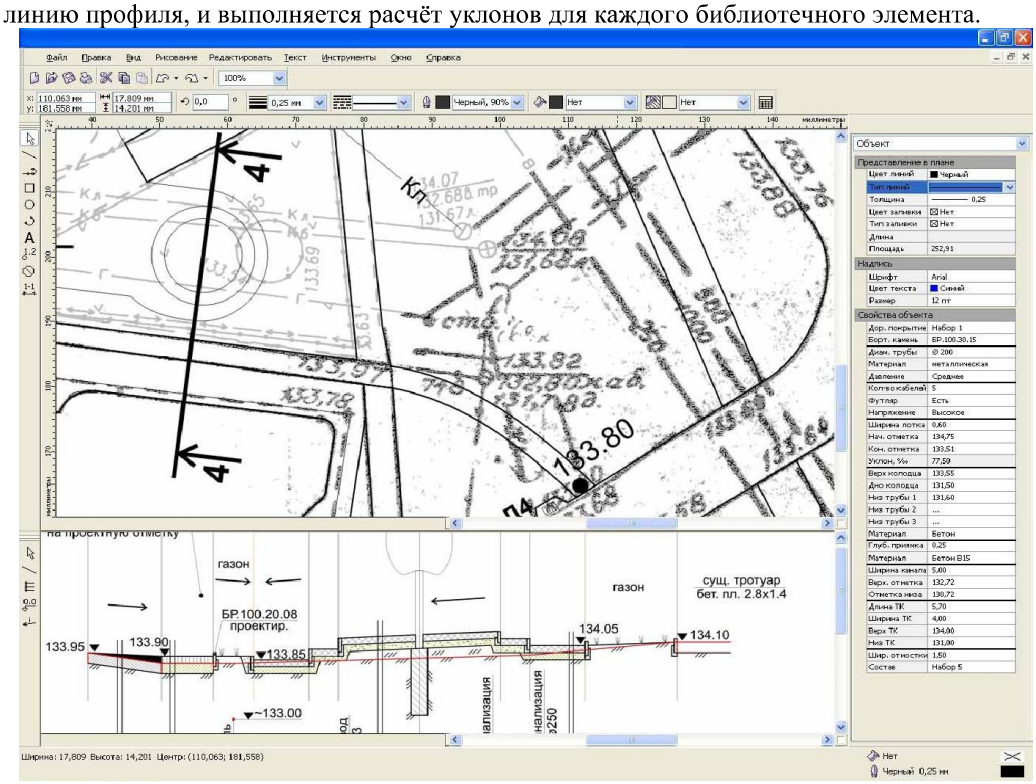

Рис. 1 Построение поперечного профиля в разработанной программе

Также для специалистов предусмотрена возможность вносить необходимые изменения и добавлять необходимую информацию непосредственно в построенный профиль. В этом случае, после автоматического построения, производится разбивка библиотечных элементов на отдельные графические примитивы.

По окончании работы с профилем у пользователя имеется возможность сохранить результаты в файл формата SVG, совместимом с большинством векторных графических редакторов.

Внедрение разработанной программы позволило уменьшить количество выполняемых пользователем однотипных операций за счёт использования настраиваемых библиотечных элементов и сократить время, затрачиваемое специалистом на проведение расчётов и графического построения рельефа поверхности на соответствующем отрезке профиля с учётом горизонтального и вертикального масштабирования.

Библиографический список

1. Bill Moore, David Dean, Anna Gerber, Gunnar Wagenknecht, Philippe Vanderheyden. Eclipse Development using the Graphical Editing Framework and the Eclipse Modeling Framework // IBM Redbooks: 2004. URL: http://www.redbooks.ibm.com/redbooks/pdfs/sg246302.pdf (дата обращения: 05.04.2013).
2. Dan Rubel, Jaime Wren, Eric Clayberg. The Eclipse Graphical Editor Framework (GEF). New Jersey, USA: Pearson Education, Inc., 2011. 267 с.
3. Vainolo. Creating a GEF editor // Can I change this later???: tutorials. 2011. URL: http://www.vainolo.com/2011/06/12/creating-a-gef-editor-part-1-defining-the-model-2 (дата обращения: 05.04.2013).
4. Васильев А.Н. Java. Объектно-ориентированное программирование. Учебное пособие. СПб.: Питер, 2013. 400 с.
5. Сорокин А., Кознов Д. Обзор проекта Eclipse Modeling Project // Сб. Системное программирование. Вып. 5, под ред. А.Н. Терехова и Д.Ю. Булычева. СПб.: Изд. СПбГУ, 2010. С. 6-31.