Н. Ю. Кульман, О. Г. Лебедев

РАЗРАБОТКА БИБЛИОТЕКИ ДЛЯ ПОСТРОЕНИЯ ГРАФА ЭЛЕКТРИЧЕСКОЙ СЕТИ

Филиал «Протвино» университета «Дубна»
Кафедра информационных технологий

Рассматривается разработка динамически подключаемой библиотеки создания графа электрической сети для комплекса SCADA «Систел» [1]. Описывается алгоритм преобразования электрической схемы в графовую модель, а также последующая работа с ней. Описываются действия разработчика при использовании алгоритмов над графом в клиентском приложении, использующем библиотеку.

Введение

Энергосистемы, включающие в себя электростанции, электрические и тепловые сети, должны работать под управлением сложных технических систем, которые предоставляют эффективный пользовательский интерфейс. ООО «Систел» занимается разработкой, проектированием и внедрением таких автоматизированных систем диспетчерского управления (АСДУ) и сбора данных, которые называются *SCADA*-системы (*Supervisory Control And Data Acquisition*).

Полученная информация от сервера *SCADA* «Систел» представляется пользователю клиентских приложений в удобном виде: — динамически изменяемых электрических схем, графиков, отчётов.

Схемы электрических сетей — это векторное графическое изображение, которое включает в себя различные элементы, являющиеся представлением реальных объектов сложных электрических систем. К таким объектам относятся: трансформаторы различной сложности, выключатели, разъединители, заземляющие ножи, шины и другие. Также на схемах изображаются кабели и линии электропередач, которые соединяют эти объекты в реальности [2].

Все элементы на схемах наделены определённым набором признаков и состояний реальных объектов. Данный набор позволяет моделировать логику поведения сети, напрямую зависящую от работы сложных электротехнических комплексов, которые они представляют.

На основе всех графических элементов, которые формируют схемы электрических сетей, можно создавать распределённые графы, которые могут решать ряд задач, связанных с мониторингом работы электрических сетей.

Основными задачами, решаемыми с помощью внедрения распределённых графовых моделей в клиентские приложения *SCADA* -систем, являются упрощение и автоматизация работы с электрическими схемами любой сложности.

Построения графа сети

Клиентские приложения, входящие в состав комплекса *SCADA* «Систел», и работающие со схемами электрических сетей, используют графическую библиотеку «Систел», в основе которой лежит интерфейс программирования приложений (*Application Programming Interface, API) GDI/GDI+ (Windows XP*) или *Direct2D (Windows 7, 8*). Клиентская часть комплекса *SCADA* «Систел» написана на языке *Visual C++* [3].

Это важно при выборе средств, которые позволят разработать модель построения распределённых графов. Для решения задачи построения графа сети был выбран пакет библиотек, который имеет прямую совместимость с языком *C++* — *Boost Libraries* [4].

Boost Libraries — это собрание множества независимых библиотек, созданных независимыми разработчиками и тщательно проверенными на различных платформах.

Для построения графов электрической сети была выбрана библиотека компонентов *Boost Graph Library (BGL*) [5].

BGL — это первая библиотека графов *C++*, применяющая понятия обобщённого программирования при создании алгоритмов на графах. Она сходна по своей организации со стандартной библиотекой шаблонов *C++* (*Standard Template Library, STL*). Для

профессиональных разработчиков *BGL* предлагает высококачественные реализации структур данных и алгоритмов.

Структуры и алгоритмы данных, представленные в данной библиотеке, позволяют решать в рамках программно-технических комплексов такие задачи на графах, как:

- поиск в ширину — обход электрической сети от заданного элемента, например, источника питания, с анализом состояния встреченных элементов;
- поиск в глубину — аналогично предыдущему алгоритму, но последовательность обхода другая;
- алгоритмы поиска кратчайшего пути — поиск от заданного узла кратчайшего пути до всех остальных, например, поиск ближайшего отсекающего устройства;
- нахождение компонент связности графа — выделение участков сети, не связанных с другими участками;
- нахождение смежных узлов.

Построение графа по схеме

Для построения успешно функционирующего распределённого графа необходимы не только различные удобные инструментарии, но и хорошая алгоритмическая составляющая.

В математической теории графов и информатике граф является совокупностью непустого множества вершин и множества рёбер, соединяющих эти вершины. Теория графов находит применение в различных областях: в информатике, программировании, логистике, экономике, коммуникационных и транспортных системах [6].

Изображение схемы электрической сети в виде графа повторяет графическое изображение схемы, но без элементов, из которых состоит электрическая цепь. Для получения графа двух- и более полюсные элементы (трансформаторы, выключатели, разъединители и др.) заменяют ветвями графа — линиями, соединяющими его узлы. Узлами электрической сети, в данном случае, являются места соединения ветвей (рис. 1).

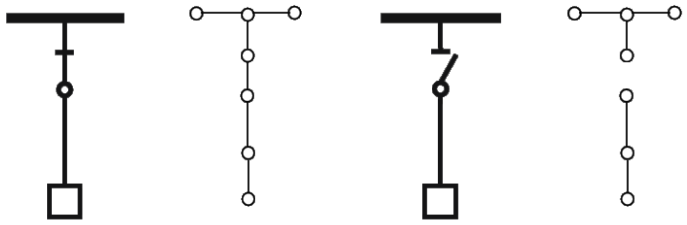

Рис. 1 Изображение участка сети и графа на его основе:
а) все элементы сети включены; *б)* разъединитель выключен

Создание графовой библиотеки и варианты работы с ней

Для реализации поставленных задач была создана динамически подключаемая библиотека (*Dynamic Link Library, DLL*), которая позволяет строить граф сети на основе изображения, работая с объектами графической библиотеки программного комплекса «Систел», а также реализует алгоритмы работы с полученным графом.

Структура библиотеки графов показана на рис. 2.

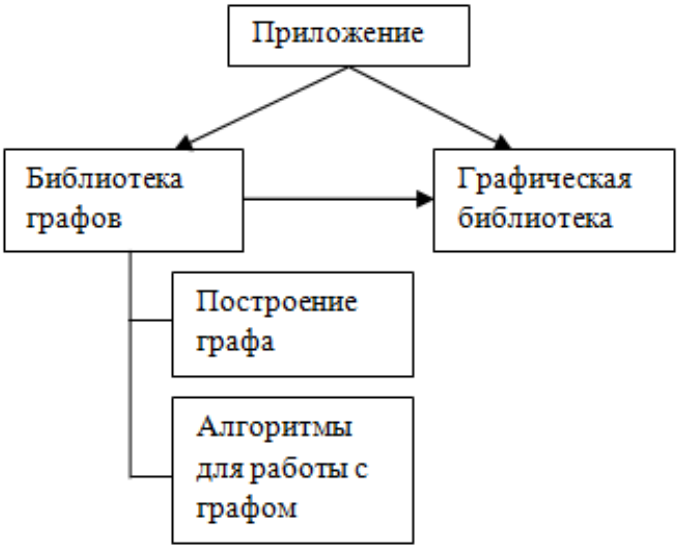

Рис. 2 Структура клиентского приложения

Например, для реализации алгоритма проверки построения схемы электрической цепи, разработчику нужно выбрать подходящий алгоритм обхода графа (в ширину или в глубину) из списка алгоритмов *BGL*, затем создать новый шаблонный класс, который наследуется от класса, реализующего алгоритм обхода в *BGL*. Внутри этого класса должны содержаться методы, которые разработчик хочет переопределить по своему усмотрению. Например, для того, чтобы отобразить правильность соединения всех элементов на схеме, можно перекрашивать в другой цвет те из них, которые соединены правильно. Эту логику необходимо реализовывать в переопределяемых методах. После этого разработчик отправляет в *DLL* экземпляр своего класса, который и будет использоваться внутри её механизмов как посетитель или событийная точка. Когда алгоритм обхода доходит до этой событийной точки, то исполнение алгоритма переходит из *DLL* в пользовательское приложение. В этой точке начинает выполняться тот алгоритм, который разработчик создал на основе переопределённых методов. После завершения этого алгоритма, управление снова возвращается в библиотеку, где она завершает весь процесс обхода графа.

Таким же образом можно эмулировать распространение тока по электрической цепи в приложениях, предназначенных для тренировки диспетчеров [7]. Идея заключается в том, что все элементы цепи должны быть правильно собраны и те элементы, которые будут обесточены, не должны быть перекрашены в другой цвет при обходе графа (рис. 3).

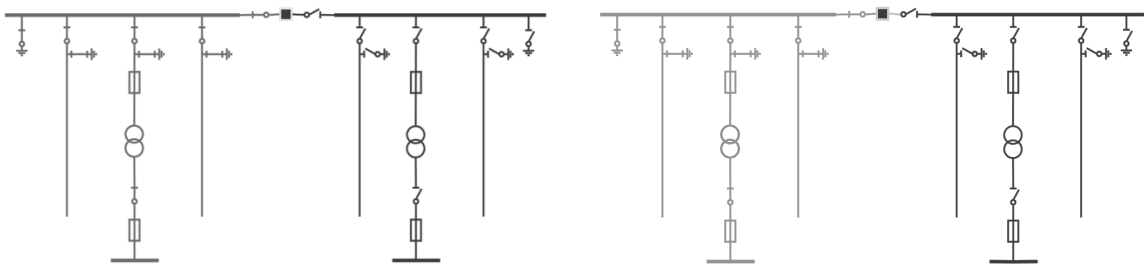

Рис. 3 Слева — собранная схема до обхода, справа — после

С помощью полученного графа сети можно будет выделить следующие понятия: клемма, физическая точка присоединения, топологический узел (рис. 4). Это позволит подготовить топологическую модель схемы, содержащую информацию о реальных объектах системы и их связях между собой.

Полученная модель, совместно с данными о реальных объектах (напряжении, состоянии и т. д.), послужит для решения различных топологических задач.

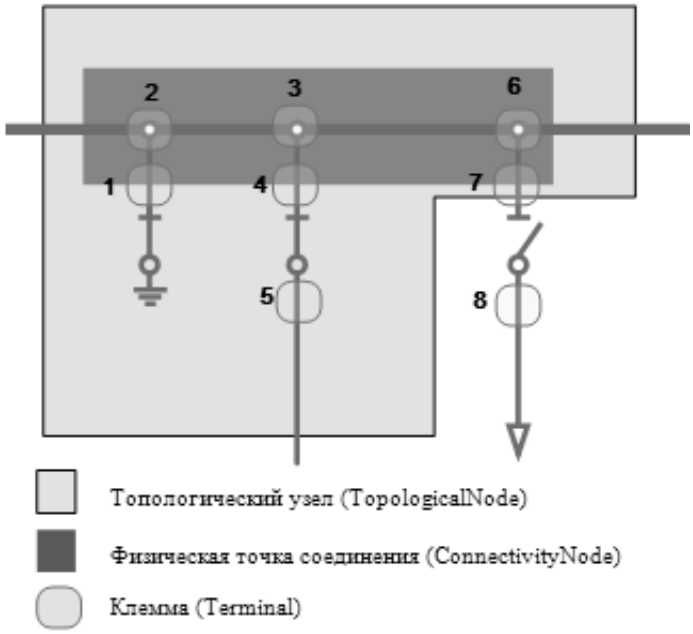

Рис. 4 Представление электрической сети
с топологической точки зрения

Библиографический список

1. Рыкованов, С. Н. Оперативный информационный управляющий комплекс «Систел» / С. Н. Рыкованов, Н. Ю. Кульман, В. И. Ухов // Автоматизация от А до Я. — 2007, Минск, №1. — С. 9—11. *(статья в зарубежном журнале)*.

2. Кульман, Н. Ю. Графический редактор в системе диспетчерского управления / Н. Ю. Кульман, А. В. Косивцова // Труды международной научной конференции MEDIAS 2011, АНО «ИФТИ».

3. Хортон, А. Visual C++ 2010: полный курс (с кодами) / А. Хортон; пер. с англ — М. : Вильямс, 2011. — 1206 с.

4. http://www.boost.org

5. Сик, Д. C++ Boost Graph Library. Библиотека программиста / Д. Сик, Л.-К. Ли, Э. Ламсдэйн; пер. с англ. Сузи Р. — Спб.: Питер, 2006. — 304 с.

6. Кормен, Т. Х. Алгоритмы: построение и анализ / Т. Х. Кормен, Ч. И. Лейзерсон, Р. Л. Ривест, К. Штайн; 2-е издание. — М.: Вильямс, 2010. — 1296 с.

7. Кульман, Н. Ю. Разработка ПО «Тренажер» для SCADA «Систел» / Н. Ю. Кульман, А. В. Косивцова // Труды конференции «Ситуационные центры и системы виртуального окружения для комплексной безопасности и антитеррористической защищенности». — 2012. С. 130–133.