УДК 519.711.3

Ю.Н. Алпатов, И.В. Борисенко*

МЕТОДИКА ЗАПИСИ ФУНКЦИОНАЛЬНОЙ СХЕМЫ СИСТЕМЫ УПРАВЛЕНИЯ ПО ТАБЛИЦЕ ВЗАИМОДЕЙСТВИЯ ПАРАМЕТРОВ

Статья посвящена проектированию автоматизированной системы идентификации многосвязных объектов управления методом структурных графов. Для разработки программы использована визуальная система программирования Microsoft Visual Studio C++ Express 2008. Рассматриваются алгоритмы построения графических моделей: функционального графа, структурной схемы, С-графа; матричных уравнений объекта управления.

Ключевые слова: многосвязный объект управления, структурная схема, структурный граф, матричное уравнение объекта управления, алгоритмы, структуры данных.

Одним из этапов построения математической модели многосвязного объекта управления методом структурных графов является запись функциональной схемы по таблице взаимодействия параметров.

Функциональная схема объекта отражает причинно–следственные отношения на множестве параметров системы и представляет собой сигнальный граф моделируемого объекта. Данная формализованная модель является теоретической основой для получения математической модели системы. С алгоритмической точки зрения на данном этапе происходит формирование структур данных, используемых далее при анализе системы управления. Вершинами графа служат параметры объекта, рёбра характеризуют наличие функциональных связей между параметрами и направление связей. Фрагмент функциональной схемы представлен на рис. 1.

Представленный в настоящей статье алгоритм был разработан в системе программирования Microsoft Visual Studio Express 2008. К достоинствам данной среды программирования следует отнести ориентированность на объектно-ориентированное программирование, наличие широкого разнообразия динамических библиотек подпрограмм, в том числе для работы с графикой, матрицами и графами, простота работы с динамическими массивами, объектами и компонентами.

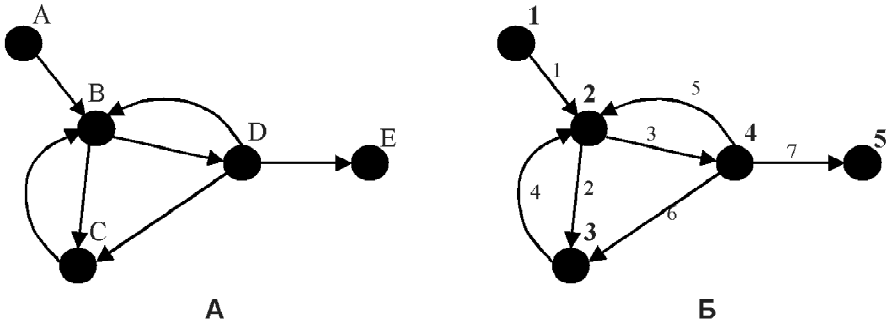

Рис. 1. Фрагмент функциональной схемы:
А – классическое представление модели, Б – идентификация элементов модели во внутреннем информационном пространстве программы.

* - автор, с которым следует вести переписку.

В графе связности выделены следующие компоненты:

1. Узел, имеющий одно свойство – имя.

2. Связь, имеющая два свойства:

- имя параметра, из которого выходит связь;
- имя параметра, в который входит связь.

Функциональная схема представлена в памяти ЭВМ в виде динамического массива **arrFS** структур **fsNode**.

```
struct fsNode   // node of functional scheme
  {
   unsigned int N; // ordinal number (порядковый номер)  N>=0
    AnsiString Name;
    int X;
    int Y;
    unsigned int c; // число инцидентных узлов
    DynamicArray <fsArc> arrOut; // в какие узлы выходит связь
  };
```

Структура fsNode содержит следующие элементы:

- N – переменная целочисленного типа. Хранит порядковый номер узла в функциональной схеме.
- Name – переменная строкового типа. Хранит имя узла функциональной схемы, отражаемое в пользовательском интерфейсе.
- X, Y – переменные целочисленного типа. Хранят координаты центра узла на плоскости.
- C – переменная целочисленного типа. Хранит количество инцидентных узлов для данного узла. С ее помощью можно определить тип узла функциональной схемы.
- ArrOut – динамический массив. Хранит массив связей, исходящих из данного узла.

Динамический массив **arrOut** содержит элементы, типа структура **fsArc**.

```
struct fsArc // arc of functional schema
  {
   unsigned int N; //  порядковый номер N>=0
    AnsiString Name;    // Arc name
    unsigned int NodeIn; // which node the arc comes in
  };
```

Структура fsArc содержит следующие элементы:

- N – переменная целочисленного типа. Содержит порядковый номер дуги, инцидентной для данного узла.
- Name – переменная строкового типа. Содержит имя дуги, отражаемое в пользовательском интерфейсе.
- NodeIn – переменная целочисленного типа. Содержит номер узла, в который входит данная связь.

Обобщенный алгоритм построения функциональной схемы по таблице взаимодействия параметров:

1. Подсчитывается количество параметров в таблице взаимодействия.
2. Инициализируется динамический массив arrFS узлов функциональной схемы с длиной, равной числу параметров в таблице взаимодействия.
3. В цикле производится обход таблицы взаимодействия параметров построчно (i), затем во вложенном цикле по столбцам (j).
4. В каждой строке подсчитывается число знаков + и сохраняется в переменную arrFS[i].c.
5. Инициализируется динамический массив arrOut для узла arrFS[i] с длиной arrFS[i].c.
6. Во вложенном цикле добавляются элементы в динамический массив arrFS[i].arrOut, соответствующие узлам, в которые выходят связи из узла arrFS[i].

Поддержание целостности данных осуществляется не только на горизонтальном, но и на вертикальном уровне. Изменения в таблице взаимодействия параметров, такие, как изменение или удаление параметра, должны приводить к соответствующим изменениям в массиве

узлов функциональной схемы, то есть к изменению или удалению узла в функциональном графе.

С точки зрения разработки пользовательского интерфейса, как видно из рис. 1, для построения графа связности необходимо знать:

- координаты каждого узла, задающие его положение относительно других узлов;
- имя параметра, представленного узлом;
- координаты концов дуг графа связности.

При создании алгоритма построения структуры функциональной схемы на основе таблицы взаимодействия параметров учитывалось прямое соответствие их элементов. Одному параметру системы соответствует один узел функциональной схемы. Одной функциональной связи между параметрами соответствует одна дуга функциональной схемы.

В алгоритме используются следующие функции и процедуры:

CircleDraw(k) – строит узлы функциональной схемы;

ArrowDraw(k) – строит дуги функциональной схемы.

Проверка функционального графа по таблице взаимодействия осуществляется по следующим правилам:

1. Количество выходных сигналов параметра равно количеству знаков «+» в строке параметра.

2. Количество входных сигналов параметра равно количеству «+» в столбце параметра.

Число узлов функциональной схемы и число дуг, связывающих узлы, может быть велико, поэтому при построении графа узлы надо расположить так, чтобы, по возможности, минимизировать число пересечений дуг. Анализ показал, что самый простой способ решения данной задачи (но не самый эффективный) – расположение узлов, имеющих большее число связанных дуг, ближе к воображаемому центру функциональной схемы, а узлов с меньшим числом связанных дуг ближе к периферии. Кроме того, узлы надо располагать таким образом, чтобы между любой парой узлов, где бы они не располагались на плоскости, можно было провести дугу так, чтобы она не прошла поверх какого-либо узла и не совпала с другой дугой.

Учитывая вышеизложенное, можно предложить следующую схему расположения узлов графа (рис. 2.).

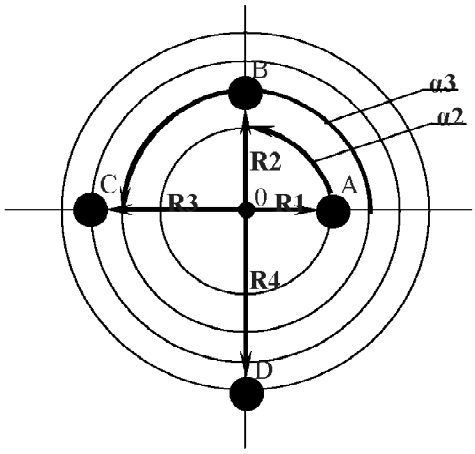

Рис. 2. Схема расположения узлов функциональной схемы.

На рис. 2. показаны четыре узла: A, B, C, D. Положение каждого узла определяется радиус-вектором; для узла A это R1, для B – R2 и так далее. Каждый радиус-вектор характеризуется определенной длиной и углом поворота относительно начала координат – точка 0. При этом длина каждого последующего радиус-вектора увеличивается на величину $dR = const$ и задается произвольно, а угол поворота увеличивается на величину $d\alpha = (2 \cdot \pi)/n$, где n – число узлов графа.

Таким образом, вычисление координат узлов графа сводится к следующим шагам:

1. Определить число узлов графа (n).

2. Задать значения следующих величин: R – начальная длина радиус-вектора, dR – приращение длины радиус-вектора, α – начальный угол поворота радиус-вектора; вычислить $d\alpha = (2 \cdot \pi)/n$

– приращение угла поворота радиус-вектора.

3. Упорядочить узлы графа в порядке убывания числа связанных с ними дуг.

4. Вычислить координаты первого узла по правилу:

$$R : R + dR, \alpha := \alpha + d\alpha, X1 := \cos(\alpha) \cdot R,$$

$$Y1 := \sin(\alpha) \cdot R7.$$

5. Повторять пункт 4 до тех пор, пока не будут вычислены координаты всех узлов.

Литература

1. Рунова Е. М. Влияние техногенного загрязнения на леса Приангарья. Братск : БрИИ, 1999. 107 с.

2. Алпатов Ю. Н. Синтез систем управления методом структурных графов. Иркутск : Изд-во ИГУ, 1988. 184 с.

3. Richter J. M., Nassare **C.** Windows via C/C++ (Pro-Developer). Microsoft Press. Redmont, Washington, 2008.

4. Bjornander S. Microsoft Visual C++ Windows Applications by Example: Code and explanation for real-world MFC C++ Applications. Birmingham, 2008.

5. Diestel R. Graph Theory. Electronic edition, 2005.

6. Макконелл Дж. Основы современных алгоритмов. 2-е изд., доп. Москва: Техносфера, 2004. 368 с.

7. CodeProject: Graph Theory: Undirected Graphs. Free source code and programming help. URL: http://www.codeproject.com. Загл. с экрана. (дата обращения : 01.05.2010.)