СПОСОБЫ ВИЗУАЛИЗАЦИИ АЛГОРИТМОВ И ПРОГРАММ

Л. Ф. Дробушевич, В. В. Конах

Белорусский государственный университет
Минск, Беларусь
E-mail: droblf@bsu.by; konakh@bsu.by

Аннотация. Рассматриваются возможности визуальной технологии программирования нового поколения в процессе обучения программированию. Обосновывается использование Р-схем как для записи алгоритмов, так и для записи программ по спецификации алгоритма.

Ключевые слова: визуальная технология программирования, графический редактор, Дракон-схема, Р-схема.

ВВЕДЕНИЕ

Кто обладает знаниями, тот и должен формализовать алгоритм этих знаний. В свое время Н. Вирт (автор языка Паскаль) утверждал, что Паскаль должен быть первым языком, с которого следует начинать изучение программирования. Это действительно так, если начинать изучение программирования с языка программирования высокого уровня. Однако сегодня ситуация изменилась: появились визуальные и графические языки, которым, по мнению многих авторов, принадлежит будущее [1]. Так как в любой программе главное – это алгоритм, то, записав его в наглядной графической форме, например, в виде Р-схемы (базис алгоритма), далее уже не принципиально операторами какого языка программирования будут нагружены дуги этого базиса. Поэтому начинать изучение алгоритмизации и программирования в системе образования (на любом уровне) целесообразно с визуальных языков.

Это утверждение можно обосновать и по-другому. Критерий хорошего понимания требует, чтобы форма записи алгоритма была максимально удобной и ясной. Чтобы создать подобную легкость, нужны понятные визуальные формы записи алгоритма. Чем понятнее запись алгоритма, тем легче найти дефекты в нем. Ведь чем больше ошибок будет обнаружено при визуальном анализе алгоритма, тем больше вероятность, что в программе их останется меньше. Кроме того, понятная визуальная форма записи алгоритма проще для объяснения сути алгоритма другому заинтересованному лицу и более удобна для изучения.

К тому же все без исключения известные языки программирования слишком сложны для обычного человека (непрофессионального программиста), чтобы записывать на них алгоритмы. Поэтому базисом может стать визуальный графический язык, каким является язык Р-схем. Любой специалист сможет выполнять алгоритмизацию своих профессиональных знаний самостоятельно, без помощи программистов. Рисовать свои мысли в виде наглядных и точных Р-схем − основа дальнейшей работы. С их помощью достаточно легко и быстро, затратив

минимум усилий на изучение, решать любые задачи, в том числе создавать программные системы.

ВИЗУАЛЬНАЯ СПЕЦИФИКАЦИЯ АЛГОРИТМОВ И ПРОГРАММ

Мысль визуализации алгоритмов сама по себе не нова. Самый первый способ визуализации алгоритмов практически появился с возникновением программирования − это старые добрые блок-схемы. И хотя эта форма записи алгоритма используется и сейчас, в частности в школьном курсе информатики, однако она не лишена некоторых недостатков. Например:

- Отсутствие явных ограничений на правила перехода в блок-схемах приводит к тому, что в них не всегда можно явно выделить базовые алгоритмические конструкции, т.е. алгоритм, записанный в виде блок-схемы, может не соответствовать требованиям структурного программирования.
- При составлении блок-схем надо внимательно следить за структурностью алгоритма, т.е. блок-схема произвольного алгоритма должна быть композицией схем основных алгоритмических конструкций, в противном случае программирование подобного алгоритма будет затруднено.
- Обозримыми являются блок-схемы только для небольших алгоритмов.
- На практике в качестве исполнителей алгоритмов используются специальные автоматы, в частности, компьютеры. Поэтому алгоритм, предназначенный для того или иного исполнителя, должен быть записан на языке, “понятном” ему, т.е. на языке программирования. Поэтому неизбежно приходится переводить графическую форму записи алгоритма в текстовую форму программы.

Известно несколько идей использования «графических языков» высокого уровня, которые непосредственно преобразуют графический алгоритм в графическую программу. Для сравнения рассмотрим два визуальных языка. Первый − это язык Дракон-схем, описанный В. Пароджановым в [2], второй − язык Р-схем, предложенный И. В. Вельбицким [3]. На Р-схемы к тому же принят стандарт ISO/IEC 8631, что немаловажно.

Дракон-схемы есть не что иное, как правильно составленные блок-схемы. Поэтому отмеченные выше недостатки блок-схем, сохраняются и в Дракон-схемах.

Графический язык Р-схем сконструирован специально, чтобы осуществить плавный переход от алгоритма, представленного в виде двумерной картинки, к двумерной программе. При этом базис алгоритма и программы не изменяется. И таким базисом является сама Р-схема, а не нагрузки её дуг.

На практике приходится сталкиваться со случаями, в которых структурное программирование может приводить к плохому выражению алгоритмической логики. Такие случаи характерны в первую очередь для алгоритмов управляющего и событийного характера (в отличие от алгоритмов информационно-преобразующего, вычислительного характера).

Рассмотрим пример 1, в котором реализуется последовательность действий, чередующаяся проверками успешности их выполнения. Результатом такой проверки могут быть два исхода — успех и отказ. Ситуация типична для многих управляющих алгоритмов. Сначала приведем фрагмент записи алгоритма на визуальном языке Дракон (рис. 1а). Далее на рис. 1б показан вариант текстовой записи алгоритма в структурном стиле, где пришлось трёхкратно продублировать действие − "отказ от пуска". И, наконец, на рис. 1в показано, как можно обойтись без дублирования путём введения явной переменной состояния.

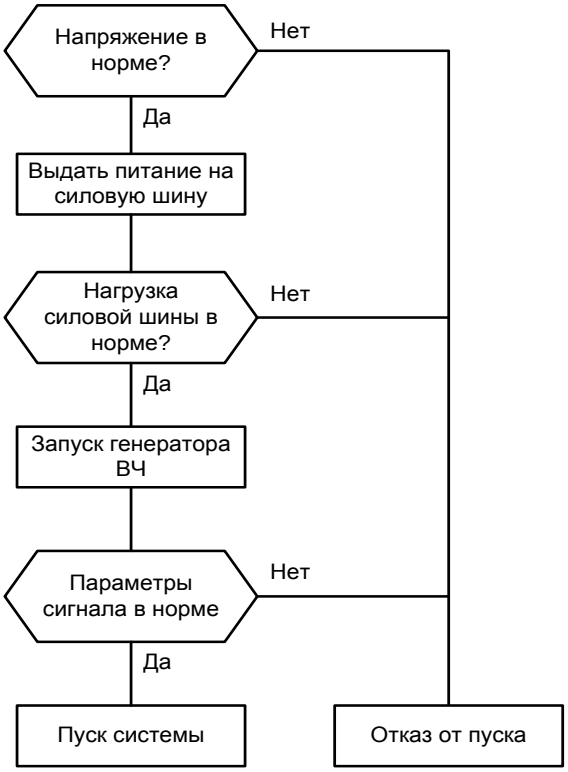

Рис. 1а. Дракон-схема

```
IF напряжение в норме THEN
   Выдать питание на силовую шину;
    IF нагрузка силовой шины в норме
    THEN
        Запуск генератора ВЧ;
         IF Параметры сигнала в норме
         THEN
             Пуск системы
         ELSE
            Отказ от пуска
         END
    ELSE
        Отказ от пуска
    END
ELSE
    Отказ от пуска
END
```

Рис.1б

```
VAR  успех: BOOLEN;
  …
успех := напряжение в норме;
IF успех THEN
    Выдать питание на силовую шину;
    успех := Нагрузка силовой шины в норме
END;
IF успех THEN
    Запуск генератора ВЧ;
    успех := Параметры сигнала в норме
END;
IF успех THEN
    Пуск системы
ELSE
    Отказ от пуска
END
```

Рис. 1в

На рис. 1г приведена запись этого же алгоритма в виде Р-схемы, где для "структуризации" будущей программы явно дублируется соответствующее действие, а на рис. 1д явно показан переход в конец алгоритма с помощью *return*, что не запрещено в рамках современного стиля программирования.

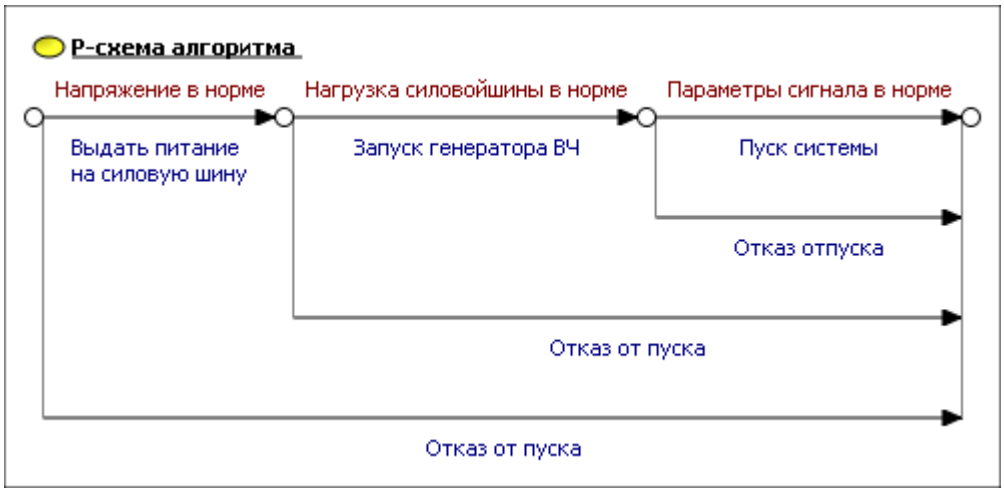

Рис. 1г. Р-схема алгоритма

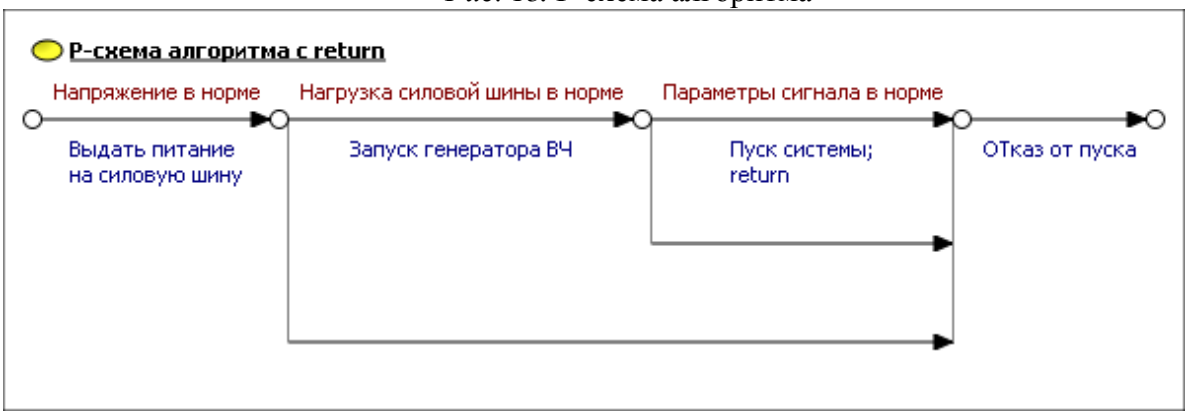

Рис. 1д. Р-схема алгоритма с использованием *return*

Второй пример связан с циклическим алгоритмом, который также имеет два исхода. Это может быть, например, алгоритм линейного поиска. Приведём сначала возможную запись алгоритма с помощью нотации Дракон-схем (рис. 2а), затем покажем текстовую запись алгоритма в классическом структурном стиле (рис. 2б), наконец приведем алгоритмически эквивалентную запись в виде Р-схемы (рис. 2в).

Дракон-схема (рис. 2а) отражает семантику происходящего: вычисление конъюнкции отрицания условий представляется алгоритмическим блоком из двух проверок с тремя исходами.

Для структуризации текстовой записи с помощью цикла while (рис. 2б) пришлось ввести дублирование проверки условия, но при этом информация о том, по причине какого условия завершился цикл, просто теряется. Маршруты вычислений сходятся в одной точке выхода из цикла, после чего необходимо вновь проверять условия.

Визуальная Р-схема (рис. 2в) показывает переход на конец цикла с помощью return. В записи явно показаны возможные исходы: первое условие истинно, второе не вычислялось; первое условие ложно, второе истинно; наконец, оба условия ложны, − происходит переход к следующему состоянию и далее − на начало проверок.

Эти исходы соответствуют принятию одного из трёх решений: искомый элемент отсутствует в последовательности; искомый элемент обнаружен и совпадает с текущим; необходимо работать в цикле и продолжить поиск.

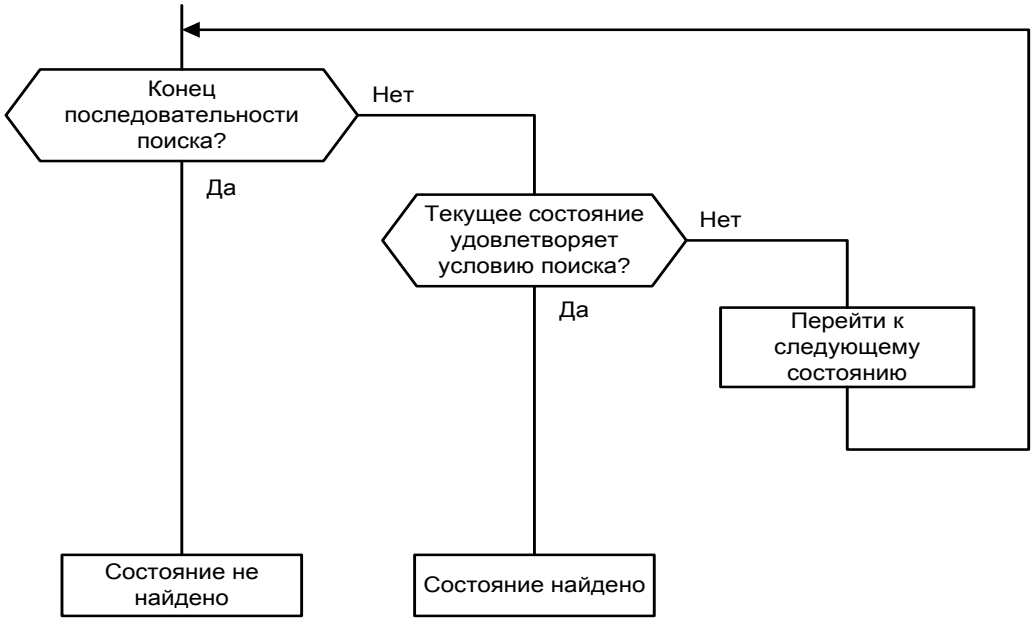

Рис. 2а

```
WHILE
    не (Конец последовательности
поиска)  & не (Текущее состояние
удовлетворяет условию поиска) DO
    Перейти к следующему состоянию
END WHILE;
IF не (Конец последовательности поиска)
THEN
    Состояние найдено
ELSE
    Состояние  не найдено
END IF
```

Рис. 2б

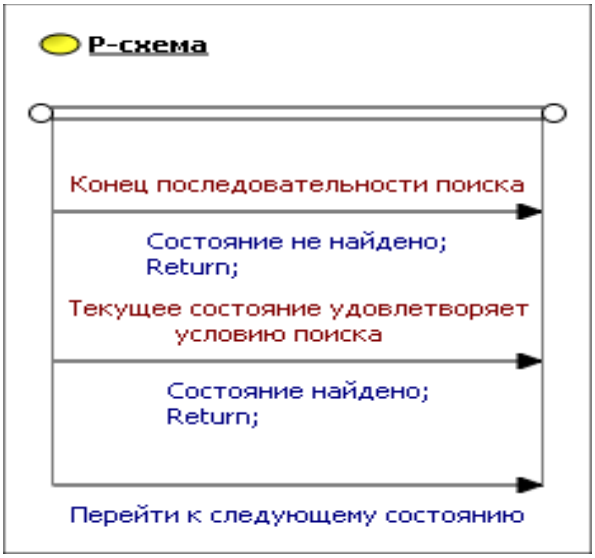

Рис. 2в

ЛИТЕРАТУРА

1. *Фаулер М., Скотт К.* UML в кратком изложении. Применение стандартного языка объектного моделирования. Пер. с англ. – М.: Мир, 1999.
2. *Пароджанов, В. Д.* Как улучшить работу ума: Алгоритмы без программистов − это очень просто! − М.: Дело. − 2001.
3. *Вельбицкий, И. В.* Визуальная технология программирования нового поколения для широкого применения на базе стандарта ISO/IEC 8631 / Межд. конф. Кипр, 2010 г.
4. *Дробушевич, Л. Ф.* Совместное использование визуальных нотаций UML и P-схем в процессе обучения объектно-ориентированным методам программирования / Л. Ф. Дробушевич, В. В. Конах // VI Межд. научн. конф. «Информационные системы и технологии» Минск, 2010.
5. *Дробушевич, Л. Ф.* Использование визуальных технологий в процессе обучения программированию. Л. Ф. Дробушевич, В. В. Конах // Информатизация образования − 2010. Межд. научн. конф. Минск, 2010.