УДК 004.4'242

DOI: 10.15827/0236-235X.120.672-677

Дата подачи статьи: 06.09.17

2017. Т. 30. № 4. С. 672–677

МЕТОДЫ И СРЕДСТВА ПОСТРОЕНИЯ АЛГОРИТМОВ ЛОГИЧЕСКОГО УПРАВЛЕНИЯ ПО ГРАФИЧЕСКИМ МОДЕЛЯМ

*А.В. Парамонов, ведущий инженер-программист, paramonov-andrew@rambler.ru
(Орский проектно-конструкторский технологический институт машиностроения,
просп. Мира, 12, г. Орск, 462403, Россия)*

В статье рассматриваются проблемы, возникающие при составлении алгоритмов управления в автоматизированных системах управления технологическими процессами на базе программируемых логических контроллеров, а именно алгоритмов логического управления. Такие алгоритмы реализуются, например, в системах управления электроприводом верхнего уровня. Процесс составления алгоритмов будем называть алгоритмизацией. Одной из основных проблем алгоритмизации задач логического управления является отсутствие визуальных, интуитивно понятных форм представления алгоритмов, не требующих от пользователя знаний сверх его компетенции. Под пользователем алгоритма здесь подразумеваются и программист, и эксперт в предметной области. В случае программиста знаниями сверх компетенции являются знания о предметной области, а в случае эксперта – знания о формальных теориях, лежащих в основе форм представления алгоритмов, навыки корректной постановки задачи управления.

Целью данного исследования является повышение уровня абстракции процессов алгоритмизации задач логического управления. Для достижения поставленной цели в статье предлагаются графические модели автоматизированных технологических установок как средство алгоритмизации указанных задач, формализуется понятие алгоритма логического управления применительно к графическим моделям, дается детальное описание методов получения по графическим моделям алгоритмов логического управления в форме таблиц решений с ограниченными входами и в форме таблиц, реализующих автоматы Мура.

Приводится описание специализированной среды разработки графических моделей и алгоритмов логического управления Logic Algorithm Designer. Описываются интерфейс программы, основные функциональные возможности, принципы работы программы. Приводятся примеры алгоритма логического управления и программного кода функционального блока МЭК 61131-3, генерируемого автоматически из алгоритма.

Предложенные в статье методы и средства алгоритмизации позволяют улучшить связь разработчика программ с экспертами предметной области, сохраняя при этом высокое качество этих программ, обеспечивая их полноту и непротиворечивость. Использование предложенных методов и средств позволяет снизить затраты времени на разработку за счет автоматического получения кода и значительно более детальной проработки алгоритма управления.

Ключевые слова: автоматизация программирования, программируемый логический контроллер, алгоритм логического управления, графическая модель, предметно-ориентированное проектирование.

Основу подавляющего большинства современных АСУ технологическими процессами (ТП) составляют программируемые логические контроллеры (ПЛК) – управляющие устройства с определенным числом входов и выходов. Принцип работы ПЛК заключается в считывании входных сигналов, выполнении над ними нужных логических и арифметических операций (возможно, с использованием внутренней памяти) и в выдаче результатов в качестве выходных сигналов. Указанная последовательность действий – считывание входов, выполнение программы и запись выходов – повторяется циклично с момента включения ПЛК и до его отключения.

Для программирования ПЛК возможно использование лишь пяти языков, стандартизированных Международной электротехнической комиссией (МЭК 61131-3) [1, 2]. К ним относятся графический язык релейно-контактных схем (Ladder Diagram) для специалистов по релейной автоматике, графический язык функциональных блок-схем (Function Block Diagram) для специалистов по полупроводниковой схемотехнике, текстовые языки – список инструкций (Instruction List) и структурированный текст (Structured Text) для программистов, а также дополнительный язык последовательных функциональных схем (Sequential Function

Chart) для структурирования организации программ.

Как известно, написанию программы, в частности для ПЛК, должно предшествовать составление алгоритма. Сузим круг рассматриваемых в данной работе алгоритмов до так называемых *алгоритмов логического управления* (АЛУ). Под АЛУ будем понимать алгоритмы, входными и выходными данными которых являются логические переменные. АЛУ может быть описан в словесной форме в терминах технологического процесса и АСУ ТП. В этом случае возникает ряд проблем, таких как потенциальная неполнота и противоречивость АЛУ, сложность проверки корректности АЛУ, сложность понимания терминов разработчиком. Более предпочтительной является ситуация, когда АЛУ описывается с помощью формальных средств, например, блок-схем алгоритмов [3, 4], таблиц решений [4–6] и тому подобных. В таком случае полнота, непротиворечивость и проверка корректности обеспечиваются формальными методами. Однако специалисту-технологу, специалисту по автоматизации, механической части, электрической части для согласования АЛУ необходимо владеть теорией, лежащей в основе использованного формального средства. Возникает необходимость разработки средств и методов составления АЛУ,

которые были бы доступны для всех участников разработки и при этом обеспечивали бы полноту, непротиворечивость и возможность проверки корректности.

В качестве таких средств в данной работе рассматриваются *графические модели* (ГМ) технологических установок и их АСУ, а также описываются методы составления АЛУ по ГМ.

Графическая модель

Для начала рассмотрим ГМ как средство для описания АЛУ [7]. ГМ с точки зрения конечного пользователя является графическим изображением технологической установки (например ленточного конвейера [8]), сколь угодно с ней визуально схожим. Степень сходства зависит от подготовленности специалиста, который будет описывать АЛУ по ГМ, от желаемого уровня удобства процесса описания. В ГМ в числе, например, деталей конструкции, механизмов, электрических шкафов показаны датчики, органы управления, средства индикации и исполнительные механизмы, а также соответствующие им сигналы. В простейшем случае в ГМ могут быть изображены только лишь входные и выходные сигналы (по отношению к АЛУ).

Формально же ГМ представляется реляционной моделью из шести отношений [9] следующим образом:

$$M = (R_E, R_{CE}, R_I, R_{EI}, R_G, R_{CG}). \quad (1)$$

На рисунке 1 представлена схема данных такой реляционной модели. Отношения R_E и R_{CE} определяют дерево элементов, из которых состоит ГМ, в котором R_E – множество вершин, а R_{CE} – множество ребер. Аналогично отношения R_G и R_{CG} представляют дерево групп. Каждый элемент относится к одной группе, что отражено в отношении R_{EI} (столбец GID – Group ID). Отношение R_I есть множество изображений, а отношение R_{EI} сопоставляет элементам изображения и определяет положение этих изображений на экране.

Рис. 1. Схема данных ГМ

Fig. 1. A graphical model data scheme

Алгоритм логического управления

ГМ в том виде, в котором она представлена выше, еще не может быть использована для составления по ней АЛУ. Для составления АЛУ в ГМ должно быть соблюдено несколько условий:

- наличие в ГМ как минимум трех групп: «входные дискретные сигналы ПЛК», «выходные дискретные сигналы ПЛК» и «остальные элементы»; элементы, относящиеся к группе «входные дискретные сигналы ПЛК», будем называть входами, а относящиеся к группе «выходные дискретные сигналы ПЛК» – выходами;
- наличие в ГМ хотя бы одного входа и хотя бы одного выхода;
- наличие в ГМ как минимум трех изображений – по одному на каждое возможное значение входа или выхода (истина, ложь, неопределенность);
- каждому входу и выходу в каждый момент времени сопоставлено только одно из этих трех изображений.

Пусть в ГМ определена операция СМЕНА(a, b), где a – номер входа или выхода; b – номер изображения значения входа или выхода. Такая операция позволяет пользователю переназначать для входа или выхода изображение. Так как каждое из трех возможных здесь изображений соответствует значению входа или выхода, операция СМЕНА позволяет управлять значениями входов и выходов.

Функция $f: A \rightarrow B$, где A – множество номеров входов и выходов; B – множество номеров изображений, ставит в соответствие каждому входу и выходу изображение значения.

Функция $g: B \rightarrow \{0, 1, \lambda\}$, где B – множество номеров изображений значений входов и выходов ($|B|=3$), ставит в соответствие каждому изображению значения само значение. Запись вида $g(f(\dot{x}_i))$, где $\dot{x}_i$ – номер входа ($i = \overline{1, n}$, n – количество входов), будем для краткости обозначать как x_i . Аналогично запись $g(f(\dot{y}_j))$, где $\dot{y}_j$ – номер выхода ($j = \overline{1, m}$, m – количество выходов), будем обозначать как y_j .

Определение 1. Ситуацией в ГМ будем называть совокупность значений всех входов и выходов и обозначать следующим образом: $s = (x_1, x_2, \dots, x_n, y_1, y_2, \dots, y_m)$.

Определение 2. Действием над ГМ будем называть совокупность значений всех выходов и обозначать следующим образом: $r = (y_1, y_2, \dots, y_m)$, причем $(\forall u \in r)[u \neq \lambda]$.

Рассмотрим небольшой пример. На рисунке 2 в ГМ два входа – A и B с номерами 1 и 2, а также два выхода – C и D с номерами 3 и 4. Ложному значению соответствует зеленый шар с цифрой 0 (изображение с номером 1), истинному – голубой шар с цифрой 1 (2), а неопределенному – серый шар с символом «-» (3).

Так как на кадре ГМ на рисунке 2 входу A соответствует голубой шар, B – серый, выходам C и D – зеленые, а $g = \{(1, 0), (2, 1), (3, \lambda)\}$, этому кадру соответствуют ситуация $s = (g(f(1)), g(f(2)), g(f(3)), g(f(4))) = (g(2), g(3), g(1), g(1)) = (1, \lambda, 0, 0)$, а также действие $r = (0, 0)$.

Применив один или несколько раз операцию СМЕНА, пользователь может изменить кадр ГМ и, таким образом, получить новую ситуацию и новое действие. Дадим формальное определение АЛУ.

Определение 3. АЛУ есть функция $F: S \rightarrow R$, где $S = \{s_1, s_2, \dots, s_k\}$, k – количество ситуаций; $R = \{r_1, r_2, \dots, r_l\}$; l – количество действий.

Следует отметить, что с учетом определения 1 АЛУ является секвенциальным, то есть текущие значения выходов зависят от значений входов и значений выходов в предыдущий момент времени. Если же необходимо определить комбинационный АЛУ, то ситуация в определении 1 переопределяется как совокупность значений только входов.

Определение 4. Кадром (t -кадром) ГМ будем называть изображение ГМ в некоторый момент времени t .

Составление АЛУ

Для того чтобы пользователь после составления АЛУ смог его в дальнейшем редактировать, необходимо обеспечить возможность получения любой пары (s, r) из F . Для этого используем *таблицу решений* (ТР) с ограниченными входами, что позволит видеть все пары (s, r) из F . В этом случае пользователь выбирает нужный для редактирования столбец в ТР, а соответствующие кадры отображаются на ГМ.

Сформулируем определение ТР следующим образом: $T = \langle X, S', Y, R' \rangle$, где $X = \{\dot{x}_1, \dot{x}_2, \dots, \dot{x}_n, \dot{y}_1, \dot{y}_2, \dots, \dot{y}_m\}$ – множество входов и выходов (только для секвенциальных АЛУ); $S' = \{s_1^T, s_2^T, \dots, s_k^T\}$ – множество вектор-столбцов ситуаций; $Y = \{\dot{y}_1, \dot{y}_2, \dots, \dot{y}_m\}$ – множе-

ство выходов; $R' = \{r_1^T, r_2^T, \dots, r_l^T\}$ – множество вектор-столбцов действий.

ТР описывает множество ситуаций $s(t)$ и соответствующих им действий $r(t+\Delta t)$, поэтому ей соответствует множество t -кадров и соответствующих им $(t+\Delta t)$ -кадров.

Пусть в ГМ определены операции:

СИТУАЦИЯ, позволяющая пользователю получить вектор $s(t)$ по t -кадру и выполнить его транспонирование $s^T(t)$;

ДЕЙСТВИЕ, позволяющая получить вектор $r(t)$ по t -кадру и выполнить его транспонирование $r^T(t)$.

Тогда последовательность действий по составлению АЛУ будет выглядеть следующим образом.

1. Выполнить операцию СМЕНА над входами и выходами $(0, n+m)$ раз).

2. Выполнить операцию СИТУАЦИЯ.

3. Выполнить операцию СМЕНА над выходами $(0, m)$ раз).

4. Выполнить операцию ДЕЙСТВИЕ.

5. Если есть неописанные ситуации, перейти к п. 1, если нет – к п. 6.

6. Конец.

Программа по АЛУ в форме ТР может быть получена методом маски, а также рядом других методов [5].

Более предпочтительным в смысле уменьшения количества операций, увеличения возможностей преобразования АЛУ в программу и упрощения проверки корректности АЛУ оказывается использование ТР, реализующих *автоматы Мура* (АМ).

Определим АМ следующим образом:

$A = \langle \hat{S}, \hat{s}_0, \tilde{S}, R, \Phi, \Psi \rangle$, где $\hat{S} = \{\hat{s}_1, \hat{s}_2, \dots, \hat{s}_p\}$ – множество внутренних состояний АМ; $\hat{s}_0 \in \hat{S}$ – начальное состояние; $\tilde{S} = \{\tilde{s}_1, \tilde{s}_2, \dots, \tilde{s}_q\}$, $q \leq k$ – множество упрощенных ситуаций в ГМ, состоящих только из значений входов; R – множество действий; $\Phi: \tilde{S} \times \hat{S} \rightarrow \hat{S}$ – функция переходов; $\Psi: \hat{S} \rightarrow R$ – функция выходов.

В случае АМ в ГМ необходимо ввести иные операции:

СОСТОЯНИЕ – позволяет пользователю получить вектор $r(t)$ и увязать его с новым состоянием $\hat{s}$, то есть добавить в Ψ пару $(\hat{s}, r(t))$;

ПЕРЕХОД – позволяет выбрать исходное состояние из $\hat{S}$, получить упрощенную ситуацию $\tilde{s}(t)$ и выбрать результирующее состояние из $\hat{S}$.

Последовательность действий по составлению АЛУ будет выглядеть следующим образом:

1. Выполнить операцию СМЕНА над выходами $(0, m)$ раз).

2. Выполнить операцию СОСТОЯНИЕ.

3. Если есть неописанные состояния, перейти к п. 1, если нет – к п. 4.

Рис. 3. Интерфейс программы Logic Algorithm Designer

Fig. 3. Logic Algorithm Designer interface

4. Выполнить операцию СМЕНА над входами ($0, n$ раз).

5. Выполнить операцию ПЕРЕХОД.

6. Если есть неописанные переходы, перейти к п. 4, если нет – к п. 7.

7. Конец.

Полученный АЛУ может быть автоматически преобразован в программу, изоморфную АМ, с помощью Switch-технологии, описанной в [4].

Программная реализация метода алгоритмизации с использованием ГМ

Предлагаемый в статье метод моделирования АЛУ совокупностью кадров ГМ реализуется в специализированной программе Logic Algorithm Designer с помощью технологий .NET и C#. Главная форма этой программы изображена на рисунке 3.

Основное пространство главной формы поделено на четыре области, обозначенные на рисунке 3 цифрами 1–4. Область 1 содержит древовидные списки всех элементов (Elements) и всех групп (Groups), а также списки всех вкладок или страниц (Pages) и алгоритмов (Algorithms).

Новым здесь является понятие страницы. Для удобства восприятия ГМ может быть разделена на

несколько страниц, каждая из которых представлена некоторым поддеревом дерева элементов. Так, например, на одной странице ГМ может располагаться визуальное изображение металлоконструкции технологической установки, на второй – изображение пульта управления, а на третьей – шкафа управления. Данная возможность обеспечивается указанием в конфигурационном файле ГМ перечня номеров групп, для элементов которых строятся отдельные страницы. Конфигурационный файл модели представляет собой XML-файл и может редактироваться с помощью как программы, так и текстового редактора.

Для возможности хранения различных АЛУ, связанных с данной ГМ, формула (1) дополняется тремя новыми отношениями:

$$M = (R_E, R_{CE}, R_I, R_{EI}, R_G, R_{CG}, R_A, R_S, R_T), \quad (2)$$

где R_A – множество алгоритмов; R_S – множество состояний автомата Мура АЛУ; R_T – множество переходов автомата Мура АЛУ.

Схема данных дополнительных отношений показана на рисунке 4.

Таблицы R_T и R_S динамически дополняются соответственно значениями входов и выходов конкретной ГМ.

Для каждого АЛУ может быть автоматически сгенерирован программный код функционального блока на языке ST по МЭК 61131-3.

В области 2 на рисунке 3 располагается инструментарий для работы с состояниями и переходами АМ выбранного АЛУ. В частности, здесь расположена кнопка «Добавить состояние», соответствующая операции СОСТОЯНИЕ, а также кнопка «Добавить переход», соответствующая операции ПЕРЕХОД.

Область 3 является основной и отображает непосредственно саму ГМ. Операция СМЕНА реализуется кликом левой кнопкой мыши по изображению сигнала. Номера изображений, соответствующих значениям сигналов, указываются в конфигурационном файле ГМ.

Область 4 содержит табличное представление АМ АЛУ. На рисунке 5 показан пример ТР АЛУ.

Название	Описание	1	2	3	4
LEFT_POS	Крайнее левое положение?			N	Y
RIGHT_POS	Крайнее правое положение?	N	Y	Y	
START	Нажата кнопка Пуск?	Y			
Стоп	Тележка неподвижна?	X		X	
Вправо	Тележка движется вправо?		X		
Влево	Тележка движется влево?				X
Стоп	Тележка неподвижна		X		X
Вправо	Тележка движется вправо	X			
Влево	Тележка движется влево			X	

Рис. 5. Пример ТР алгоритма в программе Logic Algorithm Designer

Fig. 5. An example of an algorithm decision table in Logic Algorithm Designer

На рисунке 6 показана структура файлов и папок ГМ. Здесь в папке Resources хранятся все изображения ГМ, в папке Sources – файлы сгенерированных программ. Файл с расширением *.dat представляет БД SQLite для хранения всех отношений из формулы (2). Конфигурационный файл с расширением *.mdl является основным файлом ГМ.

Код программы для АЛУ строится изоморфно соответствующему АМ с помощью конструкции

Рис. 6. Структура файлов ГМ

Fig. 6. A graphical model file structure

CASE [4]. Например, для АЛУ на рисунке 5 код будет иметь следующий вид:

```

CASE STATE OF
1: (*Стоп*)
    LRUN:=0; RRUN:=0;
    IF((NOT RIGHT_POS) AND STSRT) = TRUE THEN STATE := 2;
    ELSIF ((NOT LEFT_POS) AND RIGHT_POS) = TRUE THEN
STATE := 3;
    END_IF;
2: (*Вправо*)
    LRUN:=0; RRUN:=1;
    IF (RIGHT_POS) = TRUE THEN STATE := 1;
    END_IF;
2: (*Влево*)
    LRUN:= 1; RRUN:= 0;
    IF (LEFT_POS) = TRUE THEN STATE := 1;
    END_IF;
END_CASE;

```

Оптимизация работы с БД SQLite достигается за счет ее загрузки в оперативную память в начале сеанса работы с ГМ и выгрузки обратно в конце или при сохранении. Для этого, в свою очередь, используются классы, описывающие кортежи каждого отношения, и классы, описывающие каждое отношение в целом.

Заключение

Таким образом, в статье представлены ГМ, использующиеся в качестве средства алгоритмизации задач логического управления. ГМ являются визуально схожими с моделируемыми объектами, что делает их хорошим средством для составления АЛУ не только программистом, но и технологом, механиком, электриком, специалистом по автоматизации. В то же время АЛУ, получаемые по ГМ, являются формализованными. Данное обстоятельство позволяет применять формальные методы обнаружения и устранения неполноты и противоречивости АЛУ, в том числе и автоматически в процессе составления АЛУ, как это сделано, например, для ТР в [10]. В результате получаются полные и непротиворечивые АЛУ, по которым может быть автоматически получена программа для ПЛК.

Описанная программа Logic Algorithm Designer предоставляет средства для создания ГМ в режиме Drag and Drop, построения АЛУ и автоматической генерации программного кода на языке ST МЭК 61131-3 или Siemens SCL.

Применение реляционной модели и древовидных структур элементов и групп позволяет использовать ГМ для извлечения знаний о технологиче-

ской установке и построения экспертной системы составления АЛЮ.

Литература

1. Парр Э. Программируемые контроллеры; [пер. с англ. Б.И. Копылова]. М.: БИНОМ. Лаборатория знаний, 2007. 516 с.
2. Петров И.В. Программируемые контроллеры. Стандартные языки и приемы прикладного проектирования. М.: СОЛОН-Пресс, 2004. 256 с.
3. Захаров В.Н., Поспелов Д.А., Хазацкий В.Е. Системы управления: задание, проектирование, реализация. М.: Энергия, 1977. 424 с.
4. Шалыто А.А. SWITCH-технология. Алгоритмизация и программирование задач логического управления. СПб: Наука, 1998. 628 с.
5. Фрайтаг Г., Года В., Якоби Х., Хейнц Л., Симон Дж.,

Шпиттель У. Введение в технику работы с таблицами решений; [пер. с нем. М.Г. Гаазе-Рапопорта]. М.: Энергия, 1979. 88 с.

6. Хамби Э. Программирование таблиц решений; [пер. с англ. С.В. Диева]. М.: Мир, 1976. 88 с.
7. Парамонов А.В. Графическое моделирование технологического процесса как вспомогательное средство составления алгоритма управления // Программные продукты и системы. 2016. № 4. С. 89–93.
8. Волков Р.А., Гнутов А.Н., Дьячков В.К., Зенков Р.Л., Перетен Ю.А. Конвейеры: справочник; [под общ. ред. Ю.А. Перетена]. Л.: Машиностроение, Ленингр. отд-ние, 1984. 367 с.
9. Дейт К.Дж. Введение в системы баз данных; [пер. с англ. М.Л. Степановой]. М.: Вильямс, 2005. 1328 с.
10. Червенчук В.Д. Методы и средства синтеза алгоритмического и программного обеспечения систем управления с использованием таблиц решений: дис. ... канд. техн. наук. Омск, 1984. 123 с.

Software & Systems

DOI: 10.15827/0236-235X.120.672-677

Received 06.09.17

2017, vol. 30, no. 4, pp. 672–677

METHODS AND TOOLS OF LOGIC CONTROL ALGORITHM DESIGN USING GRAPHIC MODELS

A.V. Paramonov¹, Leading Engineer-Programmer, paramonov-andrew@rambler.ru

¹ Orsk Design Institute for Engineering Technology, Mira Ave. 12, Orsk, 462403, Russian Federation

Abstract. The paper discusses the problems of designing algorithms in automated control systems based on programmable logic controllers, in particular logic control algorithms. For example, these algorithms are implemented at the upper level of drive control systems. The algorithm designing process is called algorithmization. One of the main problems of algorithmization of logic control tasks is a lack of visual, intuitive forms of algorithms, which do not require specific knowledge from a user. Here a user of an algorithm is both a programmer and an expert in the domain. In the case of a programmer, extra knowledge is problem area knowledge, in the case of an expert extra knowledge is knowledge about the formal theories behind algorithm representation forms and the skills of correct setting of a control problem.

The aim of this research is increasing the abstraction level of logic control tasks design processes. To achieve this goal, the paper describes graphic models of automated processing stations as a tool for algorithmization of logic control tasks. It also formalizes the concept of a logic control algorithm in the context of graphic models and gives a detailed explanation of methods for obtaining logic control algorithms using graphic models in the form of the limited-entry decision tables and Moore automatic machine tables.

The paper considers specialized environment for developing graphic models and logic control algorithms called Logic Algorithm Designer. It also describes the program interface, the main functionality and working principles. There are the examples of a logic control algorithm and the program code of IEC 61131-3 function block generated automatically from the algorithm.

The proposed methods and tools of algorithmization allow improving the connection between a program developer and domain experts while maintaining high quality of these programs, ensuring their completeness and consistency. They also make it possible to reduce the development time due to automatic code generation and a much more detailed elaboration of a control algorithm.

Keywords: automatic programming, programmable logic controller, logic control algorithm, graphic model, domain-driven design.

References

1. Parr E.A. *Programmable Controllers. An Engineer's Guide*. 3rd ed., Newnes Publ., 2003, 448 p. (Russ. ed.: Moscow, Binom. Laboratoriya znany Publ., 2007, 516 p.).
2. Petrov I.V. *Programmiruemye kontrollery. Standartnye yazyki i priemy prikladnogo proektirovaniya* [Programmable Controllers. Standard Languages and Techniques for Applied Design]. Moscow, SOLON-Press, 2004, 256 p.
3. Zakharov V.N., Pospelov D.A., Khazatsky V.E. *Sistemy upravleniya. Zadanie. Proektirovanie. Realizatsiya* [Control Systems. Setting. Designing. Realization]. Moscow, Energiya Publ., 1977, 424 p.
4. Shalyto A.A. *SWITCH-tehnologiya. Algoritimizatsiya i programirovanie zadach logicheskogo upravleniya* [SWITCH-technology: Algorithmization and Programming of Logical Control Problems]. St. Petersburg, Nauka Publ., 1998, 628 p.
5. Freitag G., Gode W., Jacobi H., Lautz H., Simon J., Spittel U. *Einführung in die Entscheidungstabellentechnik* [Introduction to the Technique of Working with Decision Tables]. Berlin, Verlag Technik Publ., 1976, 88 p. (Russ. ed.: Moscow, Energiya Publ., 1979, 88 p.).
6. Humby E. *Programs from Decision Tables*. Macdonald and Co. Publ., 1973, 91 p. (Russ. ed.: Moscow, Mir Publ., 1976, 88 p.).
7. Paramonov A.V. Graphical modeling of technological process as a support tool for designing of control algorithm. *Programmye produkty i systemy* [Software & Systems]. Tver, 2016, vol. 4, pp. 89–93 (in Russ.).
8. Volkov R.A., Gnutov A.N., Dyachkov V.K., Zenkov R.L., Pereten Yu.A. *Konveyery: spravochnik* [Conveyors: Manual]. Leningrad, Mashinostroenie Publ., 1984, 367 p.
9. Date C.J. *An Introduction to Database Systems*. 8th ed., Addison Wesley Publ., 2004, 1336 p. (Russ. ed.: Moscow, Vilyams Publ., 2005, 1328 p.).
10. Chervenчук V.D. *Metody i sredstva sinteza algoritmicheskogo i programmnogo obespecheniya sistem upravleniya s ispolzovaniem tablits resheny* [Methods and Tools of Algorithms and Software Synthesis for Control Systems Using Decision Tables]. Ph.D. Thesis, Omsk, 1984, 123 p.