УДК 681.5
DOI 10.21685/2072-3059-2016-2-2

В. Н. Дубинин, Д. А. Будаговский, Д. Н. Дроздов, Д. В. Артамонов

ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ СИСТЕМ УПРАВЛЕНИЯ ДИСКРЕТНЫМИ СОБЫТИЙНЫМИ СИСТЕМАМИ НА ОСНОВЕ ИЕРАРХИЧЕСКИХ МОДУЛЬНЫХ НЕДЕТЕРМИНИРОВАННЫХ АВТОМАТОВ (Ч. 2. МЕТОДЫ И СРЕДСТВА)[1]

Аннотация.

Актуальность и цели. Объектом исследования являются системы управления для дискретных событийных систем (ДСС). Предмет исследования – методы и средства проектирования систем управления ДСС на основе автоматных моделей. Цель – разработка методов и средств проектирования и реализации систем управления ДСС на основе программируемых логических контроллеров с использованием иерархических модульных недетерминированных автоматов (ИМНДА), являющихся развитием концепции недетерминированных автоматов, предложенных Н. П. Вашкевичем.

Материалы и методы. Для решения поставленных задач использовались методы теории множеств, конечных автоматов, сетей Петри, сетевых систем «условие – событие» (NCES-сетей), а также методы разработки программного обеспечения управляющих систем на основе программируемых логических контроллеров.

Результаты. Разработаны методики реализации базисного модуля ИМНДА на основе релейно-контактной логики (LD) и составного модуля ИМНДА на основе функционально-блоковых диаграмм (FBD). Предложена структура инструментальных средств для поддержки проектирования и реализации систем управления ДСС на основе ИМНДА. Показана применимость предложенных методов на примере разработки простой системы управления.

Выводы. Предложенные методы и средства позволяют: уменьшить затраты на разработку, модификацию и сопровождение систем управления ДСС; увеличить степень повторного использования разработанных артефактов проектирования (например, автоматных модулей); автоматизировать разработку систем управления ДСС, что, в конечном счете, приводит к сокращению сроков проектирования систем управления ДСС.

Ключевые слова: дискретные событийные системы, недетерминированные автоматы, управление, модуль, проектирование, реализация, инструментальная система, программируемый логический контроллер, релейно-контактная логика, функционально-блоковые диаграммы.

V. N. Dubinin, D. A. Budagovskiy, D. N. Drozdov, D. V. Artamonov

DESIGN AND IMPLEMENTATION OF DISCRETE EVENT DYNAMIC SYSTEMS' CONTROL BASED ON HIERARCHICAL MODULAR NONDETERMINISTIC AUTOMATA (PART 2. METHODS AND TOOLS)

[1] Исследование выполнено в Пензенском государственном университете за счет гранта Российского научного фонда (проект №151110010).

Abstarct.

Background. The object of the research is control of discrete event dynamic systems (DEDS). The subject of the research is methods and tools for design of DEDS control on the basis of automata models. The goal of this work is to develop methods and tools for design and implementation of DEDS control based on programmable logical controllers (PLC) using hierarchical modular nondeterministic automata (HMNA) which are an extension of the Vashkevich's nondeterministic automata concept.

Materials and methods. To achieve the above-mentioned objectives, the methods of set theory, automata theory, Petri nets, net condition-event systems (NCES) as well as methods of PLC software development were used.

Results. The paper proposes an approach to basic HMNA modules implementation using IEC 61131-3 ladder diagrams (LD) and an approach to composite HMNA modules implementation using function block diagrams (FBD). The structure of tools to support design and implementation of HMNA-based DEDS control is suggested. And finally, the paper gives an example to demonstrate the proposed approaches.

Conclusions. The proposed methods and tools allow to: 1) reduce the cost of development, modification and maintenance of DEDS control; 2) increase the degree of re-using of design artifacts (e.g., automata modules); 3) to automate the development of DEDS control that in turn eventually leads to shortening of a period of DEDS control design.

Key words: discrete event dynamic systems, nondeterministic automata, control, module, design, implementation, tools, programmable logic controller, ladder diagram, function block diagram.

Введение

Наиболее часто в качестве аппаратной платформы для реализации управления дискретными событийными системами (ДСС) [1], к которым, в частности, относятся многие объекты из сферы промышленной автоматизации, используются программируемые логические контроллеры (ПЛК) [2]. Их особенностью является циклический характер функционирования. Стандарт IEC 61131-3 определяет пять языков программирования ПЛК (*LD*, *FBD*, *SFC*, *ST*, *IL*), первые три из которых являются графическими [3]. Хороший обзор и анализ использования формальных методов в программировании ПЛК дан в работе [4].

Автоматный подход к проектированию систем управления ДСС с ориентацией на разработку программ ПЛК является одним из основных и постоянно совершенствуется, что видно по публикационной активности в данной области. Одной из первых в 1991 г. появилась *SWITCH*-технология [5]. В работе [6] введен новый класс автоматов (*PLC-Automata*), специально предназначенный для моделирования ПЛК с учетом временных свойств. В статье [7] представлена и проиллюстрирована методология проектирования логических контроллеров на основе модульных автоматов. Работа [8] предлагает методологию моделирования, верификации и генерации кода для ПЛК на основе использования расширенной модели конечного автомата (*Timed-MPSG*), включающей сообщения. При этом для верификации используется метод *Model checking* и язык *SMV*. В статье [9] описывается синтаксис и формальная семантика нового класса временных автоматов, которые предназначены для моделирования поведения систем реального времени. Разработан формаль-

ный метод для автоматической генерации программ ПЛК на основе предложенной модели. Исследование [10] представляет методологию моделирования и валидации управляющих программ промышленных систем с последовательными, параллельными и временными операторами с использованием формализма, основанного на диаграмме состояний – «базисные диаграммы состояний (*BSC*)». В статье [11] конечные автоматы использовались для перепроектирования программ ПЛК. В работе [12] была определена модель иерархических модульных недетерминированных автоматов (ИМНДА), являющихся расширением недетерминированных автоматов (НДА), предложенных Н. П. Вашкевичем [13], которая ориентирована на формальное описание параллельных систем управления ДСС. Кроме того, существует ряд работ [11, 14], посвященных использованию языка *UML* (включая диаграммы *UML state chart*) в проектировании систем управления на базе ПЛК. В настоящее время ряд коммерческих программных продуктов поддерживает автоматный подход к проектированию программ ПЛК. Например, программный пакет *Simulink PLC Coder* фирмы *MathWorks* позволяет преобразовывать автоматную модель (*Stateflow*) в программу ПЛК на языке *ST* [15].

Несмотря на значительный прогресс в области проектирования управляющих систем на основе автоматной модели и ПЛК, ряду вопросов в литературе было уделено мало внимания. В частности, недостаточно рассмотрены вопросы реализации на ПЛК автоматных моделей, допускающих как параллелизм, так и модульное (структурное) проектирование. Модель ИМНДА покрывает эти аспекты в полном объеме. Кроме того, в большинстве рассмотренных случаев отсутствует комплексная инструментальная поддержка автоматной методологии проектирования. Все перечисленные аспекты определяют актуальность настоящей работы, нацеленной на развитие автоматного подхода к проектированию и реализации параллельных управляющих систем. Для реализации ИМНДА выбраны два графических языка ПЛК: язык релейно-контактных схем (язык *LD*) и язык функционально-блоковых диаграмм (язык *FBD*).

1. Реализация базисного модуля ИМНДА на языках *LD* и *FBD*

Недетерминированные автоматы (НДА), определенные в [13], нашли практическое применение в проектировании аппаратных средств вычислительной техники и систем управления. Одной из основных особенностей НДА является то, что все разрешенные переходы в НДА совершаются параллельно и без конфликтов. Для целей реализации с каждым состоянием НДА будем связывать битовый элемент памяти, хранящим его статус. Для простоты считаем, что состояние и реализующий его элемент памяти имеют одинаковые имена. Если $S_i = 1$, то НДА находится в состоянии S_i, если $S_i = 0$, то нет. Назовем *входо-выходным* состоянием НДА состояние, в которое входит хоть одна дуга и одновременно выходит хотя бы одна дуга. Основная проблема при реализации НДА заключается в том, что возможны конфликты в элементах памяти при одновременном выполнении над ними нескольких параллельных операций (операций чтения и записи). Для решения этой проблемы предлагается двухтактная схема выполнения НДА. Вводится понятие «двухкомпонентного» состояния. На первой фазе выполнения НДА синхронно производятся разрешенные (обычные) переходы состояний НДА, при этом уста-

навливаются первые компоненты состояний на основе вторых компонентов состояний. Таким образом, вторые компоненты состояний являются действующими, а первые – буферными. Первая фаза является основной. Суть второй фазы связана с актуализацией вторых (основных) компонентов состояний на основе первых (буферных). Вторая фаза является вспомогательной.

На рис. 1 представлено правило расщепления состояния S на два компонента – S' и S''. На данном рисунке сплошными дугами показаны переходы, выполняемые на первой фазе. Пунктирной линией между подсостояниями отмечен переход, выполняемый на второй фазе.

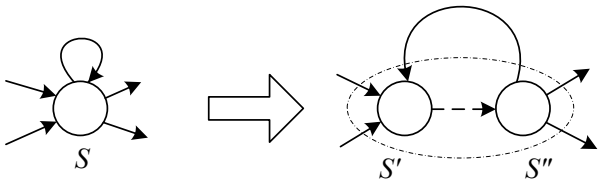


Рис. 1. Правило расщепления состояния

На фазе 1 формула для определения активности первых компонентов состояний следующая:

$$S'_j = \bigvee_{S''_i \in pre(S'_j)} (S''_i \wedge C_{i,j}), \quad (1)$$

где $C_{i,j}$ – условия разрешенности перехода из i-го состояния в j-е состояние. Функция *pre* используется для определения множества предшествующих состояний. Вторая фаза описывается выражением

$$S''_j = S'_j. \quad (2)$$

В соответствии с данной формулой активность второго компонента состояния полагается равной активности соответствующего первого компонента.

При реализации НДА на языке *LD* используется концепция реле для моделирования элементов памяти, представляющих состояния. В соответствии с двухкомпонентным строением состояния реализационной формы НДА будем выделять реле первой ступени для реализации первых компонентов состояний и реле второй ступени соответственно для реализации вторых компонентов.

На рис. 2 приведена общая схема реализации базисного модуля ИМНДА на основе комбинирования языков *LD* и *FBD*.

Как видно из рис. 2, автоматный модуль реализуется в виде двух функциональных блоков (ФБ): *Fmain* и *Fout*. Первый ФБ реализует основную логику автоматного модуля, в то время как второй ФБ выполняет формирование выходных информационных («уровневых») сигналов. Выделение выходной логики автоматного модуля в отдельный ФБ (выполняемый после всех ФБ типа *Fmain* в системе) связано с тем, что формирование актуальных информационных сигналов (автомата Мура) необходимо выполнять не на уровне отдельного модуля, а в рамках специальной единой фазы на уровне

всей подсистемы. Это связано с требованиями правильной проводки как событийных («мгновенных»), так информационных («уровневых») сигналов между модулями. Следует заметить, что информационные связи между модулями могут образовывать циклы, в то время как топология сети из событийных связей в общем случае есть ациклический граф.

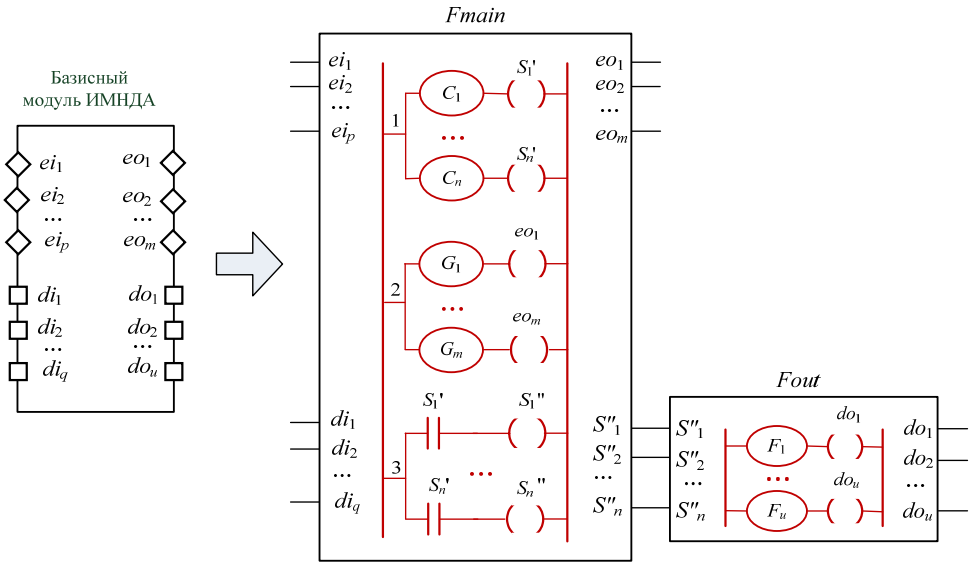


Рис. 2. Общая схема реализации базисного модуля ИМНДА

Лестничная диаграмма, инкапсулированная в ФБ *Fmain*, содержит три группы цепей. Первая группа цепей реализует первую фазу выполнения НДА. Макроцепи, вычисляющие условия возбуждения реле $C_i (i = \overline{1,n})$, обозначены на рис. 2 овалами. Вычисление этих условий выполняется в соответствии с формулой (1). Вторая группа цепей реализует вычисление выходных значений событийных («импульсных») сигналов. Условия $G_k (k = \overline{1,m})$ вычисляются следующим образом:

$$G_k = \bigvee_{(S_i, S_j) \in E_k} (S_i'' \wedge C_{i,j}),$$

где E_k – множество переходов автомата, маркированных событийным сигналом eo_k.

Третья группа цепей предназначена для реализации второй фазы выполнения НДА (2). Условие разрешенности перехода из одного состояния НДА в другое определяется в виде конъюнкции входных сигналов и состояний, взятых с отрицанием и без [13]. Пример кодирования условия $ei_1 \wedge \sim di_2 \wedge$ $\wedge\, di_3 \wedge S_1 \wedge \sim S_2$ на языке *LD* приведен на рис. 3.

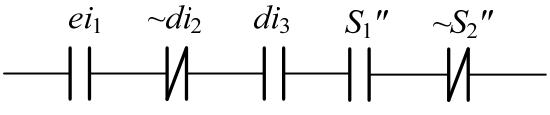


Рис. 3. Пример условия разрешенности перехода на языке *LD*

Лестничная диаграмма, инкапсулированная в ФБ *Fout*, вычисляет значения информационных выходных сигналов на основе вторых компонентов состояний. Используемые макроцепи $F_k(k=\overline{1,u})$ в виде параллельно соединенных контактов вычисляют функции

$$F_k = \bigvee_{S_i \in Z_k} S_i'',$$

где Z_k – множество состояний, при активных значениях которых выдается информационный сигнал do_k.

Следует заметить, что в определенных случаях базисный модуль ИМНДА может быть реализован в виде одного ФБ. При этом диаграммы *LD* модулей *Fmain* и *Fout* сливаются. Условием такого объединения является возможность комбинированной проводки как событийных, так и информационных сигналов (с использованием алгоритма, представленного в разд. 2).

2. Реализация составного модуля ИМНДА на языке *FBD*

В данном разделе приводится метод реализации составных модулей ИМНДА в виде функционально-блоковых диаграмм (*FBD*). Каждый (внутренний) модуль составного модуля ИМНДА реализуется одним ФБ. Основной трудностью является преодоление семантического разрыва между параллельным функционированием модели ИМНДА и последовательным (слева направо и сверху вниз) выполнением диаграмм *FBD*. Иным способом данную задачу можно сформулировать как моделирование работы ИМНДА с использованием *FBD*.

Основным принципом, положенным в основу метода, является возможность выполнения модуля только после выполнения всех «предшествующих» ему (по событийным связям) модулей. Это гарантирует правильное распространение мгновенных сигналов по сети из событийных связей, определенное в формальной модели ИМНДА [12]. Данный принцип является избыточным в том плане, что учитывает все возможные варианты прохождения мгновенных сигналов по сети из событийных связей.

Для реализации указанного выше принципа предлагается алгоритм, осуществляющий проводку мгновенных сигналов в ИМНДА с одновременным отображением модулей по *FBD*-цепям. С каждым модулем связывается пороговая константа, равная числу входящих событийных связей из других модулей, а также счетчик числа принятых мгновенных сигналов. Источником мгновенных сигналов в составном ФБ являются импульсные входы интерфейса. Внутренние модули, связанные с ними, должны выполняться первыми. После выполнения модуля производится рассылка мгновенных сигналов соседним модулям-последователям, значение счетчиков последних инкрементируется. При достижении значения счетчика пороговой величины модуль считается выполненным и включается в текущую *FBD*-цепь. Полное описание алгоритма имеет следующий вид:

```
1:    foreach m ∈ M do
2:        c(m) = 0 ;
3:        r(m) = n(m) − n_F(m) ;
4:    end foreach;
```

```
5:   select  m_L from {m ∈ M | r(m) = 0} ;  //произвольный выбор
                                            //первого модуля цепи
6:       k = 1 ;
7:       Q_k = {m_L} ;
8:       M = M − {m_L} ;
9:       while  M ≠ ∅  do
10:         found = false;
11:         M_C = succ(m_L)
12:         foreach  m ∈ M_C do  c(m) = c(m) + 1  end foreach;
13:         foreach  m ∈ M_C do // поиск модуля
    //с достигнутым максимальным значением счетчика на M_C.
14:            if  c(m) == r(m)  then
15:               Q_k = Q_k ∪ {m};
16:               m_L = m;
17:               found = true;
18:               break;
19:            end if;
20:         end foreach;
21:         if found == false then //построение новой цепи
22:            foreach  m ∈ M − M_C do // поиск модуля
               // с достигнутым максимальным значением
               // счетчика на множестве «остальных» модулей
23:               if  c(m) == r(m)  then
24:                  k = k + 1 ;
25:                  Q_k = {m};
26:                  m_L = m;
27:                  break;
28:               end if;
29:            end foreach;
30:         end if;
31:         M = M − {m_L} ;
32:      end while;
```

В приведенном алгоритме приняты следующие обозначения: $c(m)$ – значение счетчика модуля m; $r(m)$ – пороговая константа модуля m; $n(m)$ – общее количество импульсных входов модуля m; $n_F(m)$ – количество импульсных входов модуля m, непосредственно связанных с интерфейсом; k – номер текущей *FBD*-цепи; M – множество еще не распределенных по цепям модулей (первоначально включает все модули); m_L – последний добавленный в цепь модуль; M_C – множество модулей-последователей, связанных событийными связями с последним добавленным в цепь модулем; Q_k – формируемые цепи модулей; *found* – флаг, устанавливающийся, если очередной модуль был выбран из M_C; *succ*(m) – множество модулей-последователей, принимающих сигналы с модуля m.

На рис. 4 приведен пример составного модуля ИМНДА. В табл. 1 представлены реализующие его *FBD*-цепи, полученные в результате работы вышеприведенного алгоритма.

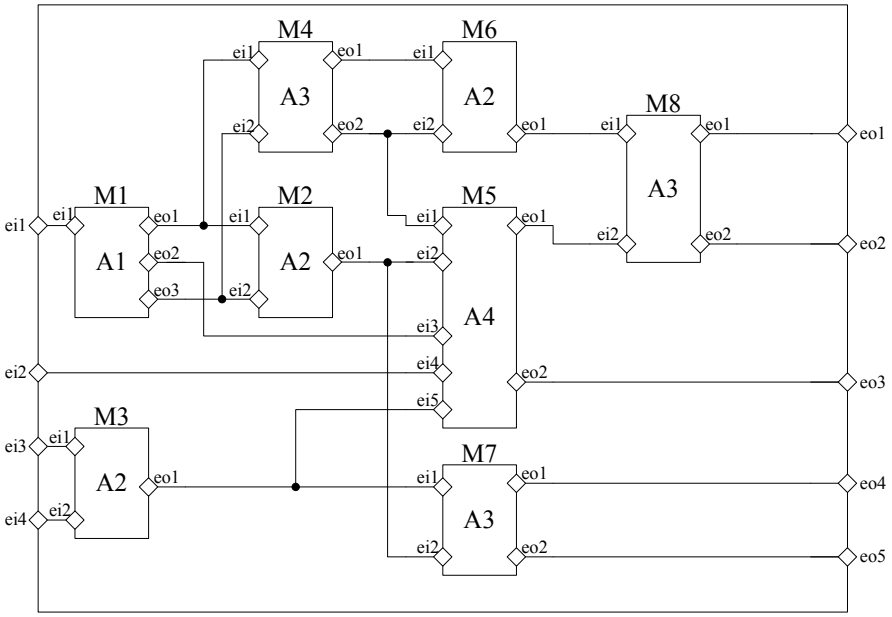
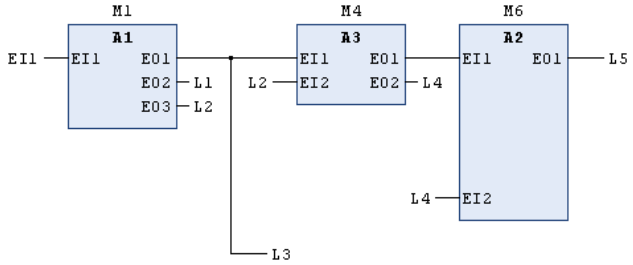
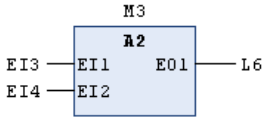
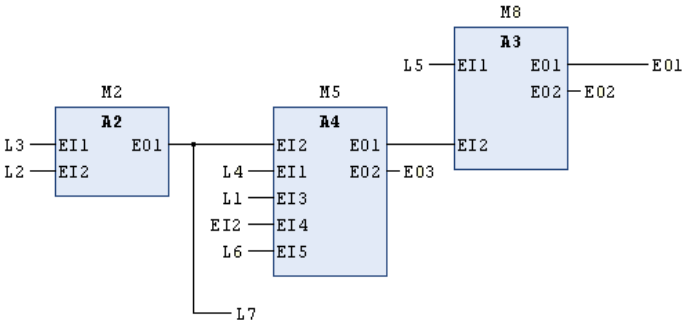
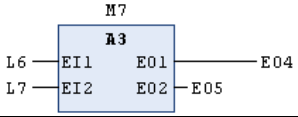


Рис. 4. Пример составного модуля ИМНДА

Таблица 1

Результирующие *FBD*-цепи

Цепь 1	
Цепь 2	
Цепь 3	
Цепь 4	

3. Инструментальные средства для поддержки проектирования и реализации систем управления ДСС на основе ИМНДА

Для проектирования систем управления ДСС на основе ИМНДА предлагается соответствующая инструментальная система поддержки проектирования. Ее структура, включающая также взаимосвязи с внешними программами (которые отмечены пунктирными прямоугольниками), приведена на рис. 5. Для формирования базисного модуля ИМНДА используется специально разработанный графический редактор *ACAD*, а для формирования составного модуля ИМНДА – графический редактор системы *ViVe* [16]. Несмотря на то, что система *ViVe* ориентирована на моделирование *NCES*-сетей (*Net Condition/Event Systems*), схожесть составных модулей ИМНДА и *NCES*-сетей позволяет использовать графический редактор этой системы.

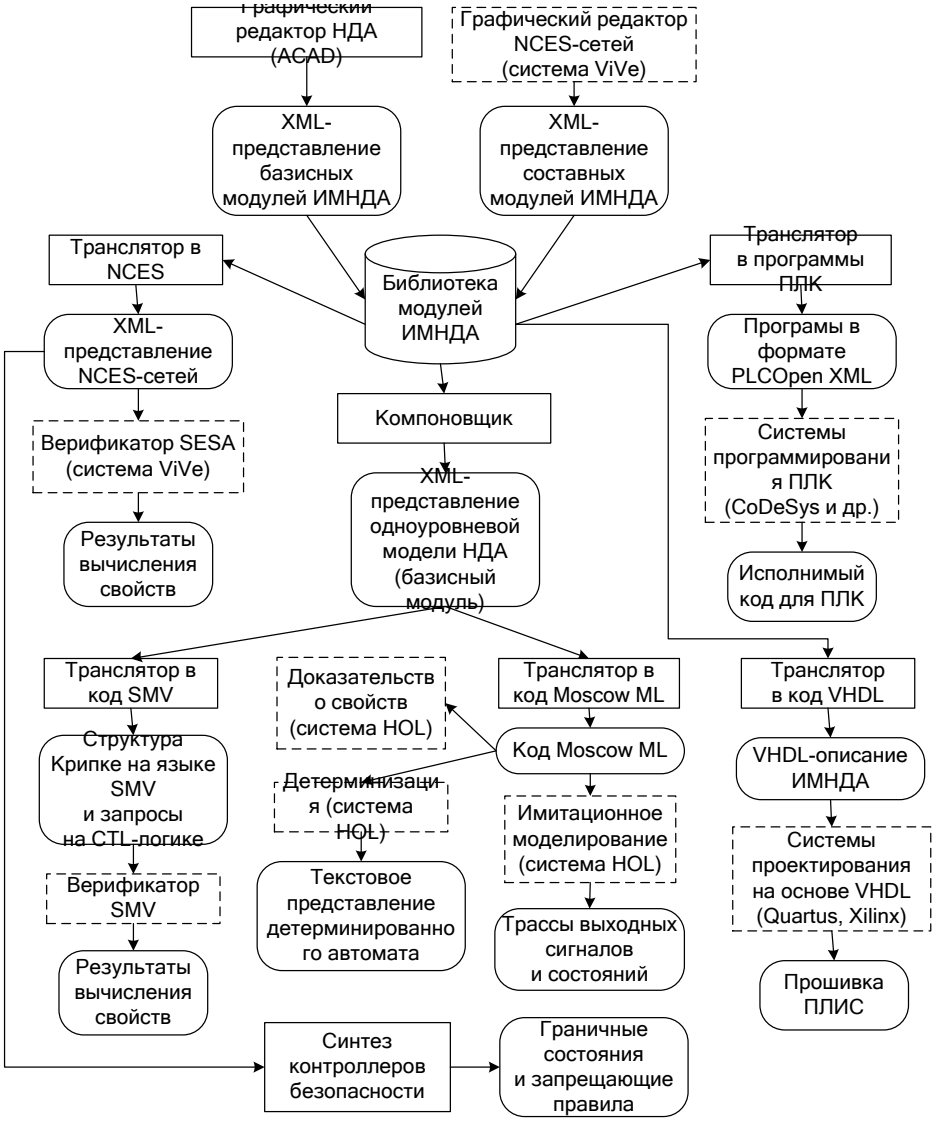


Рис. 5. Структура системы поддержки проектирования и реализации управления ДСС на основе ИМНДА

Компоновщик предназначен для генерации одноуровневой модели НДА из иерархической модели, включающей множество базисных и составных модулей. Иерархическая модель строится на основе типизации модулей. При этом каждый модуль в модели характеризуется именем типа и именем экземпляра. *Верификация* ИМНДА в системе ориентируется на метод *Model Checking*, причем реализовано два способа для осуществления этого типа верификации – с использованием верификаторов *SMV* и *SESA*. Первый способ основан на преобразовании ИМНДА в код *SMV*. В верификаторе *SMV* свойства ИМНДА формулируются в виде формул временной логики *CTL*. Второй способ основан на преобразовании ИМНДА в *NCES*-сети и использовании верификатора *SESA*, входящего в систему *ViVe*. Для детерминизации [13] и имитационного моделирования ИМНДА используется система доказательства теорем *HOL* (вернее, подсистема функционального программирования на языке *MoscowML*). *XML*-описание базисного модуля ИМНДА транслируется в данные для программы детерминизации НДА, а также в программы на языке *MoscowML* для тестирования. Транслятор *XML*-описаний ИМНДА в код *VHDL* предназначен для автоматического преобразования базисных и составных модулей ИМНДА в соответствующие описания на языке *VHDL*. Транслятор *XML*-описаний ИМНДА в программы для ПЛК предназначен для автоматического преобразования ИМНДА в программы ПЛК, представленные в соответствии со стандартом *PLCOpen XML*. Транслятор работает в соответствии с методами и алгоритмами, представленными в данной работе. Подсистема синтеза контроллеров безопасности предназначена для генерации запрещающих правил, предотвращающих переход системы в опасное состояние.

Следует также отметить, что пять компонентов предложенной инструментальной системы зарегистрированы в фондах алгоритмов и программ (Роспатент, ОФЭРНиО).

4. Демонстрационный пример

Схематично покажем применимость предложенных методов на простом примере проектирования системы управления для технологической установки, предназначенной для приготовления раствора (рис. 6). В аппарат 1 объемом 100 л подается 20 л компонента А, затем 80 л компонента В, после чего вся смесь перемешивается в течение 20 мин. В аппараты 2 и 3 объемом 100 л каждый подается приготовленная в аппарате 1 смесь (А+В), которая в дальнейшем подогревается до температуры 50 °С в аппарате 2 и до 80 °С в аппарате 3. Приготовленные растворы выгружаются для дальнейшего использования.

Входные сигналы системы управления приведены в табл. 2.

ИМНДА для управления технологической установкой на рис. 5 представлен на рис. 7.

Реализация составного модуля ИМНДА на языке *FBD* (в системе *CoDeSys*) представлена в табл. 3, а реализация базисного модуля А2 в виде диаграммы *LD* – на рис. 7. Следует заметить, что в данном примере использовалась одноблочная реализация базисных модулей ИМНДА, что объясняется тем, что удалось найти такой порядок выполнения ФБ, который обеспечивает правильную проводку событийных и информационных сигналов при данном виде реализации.

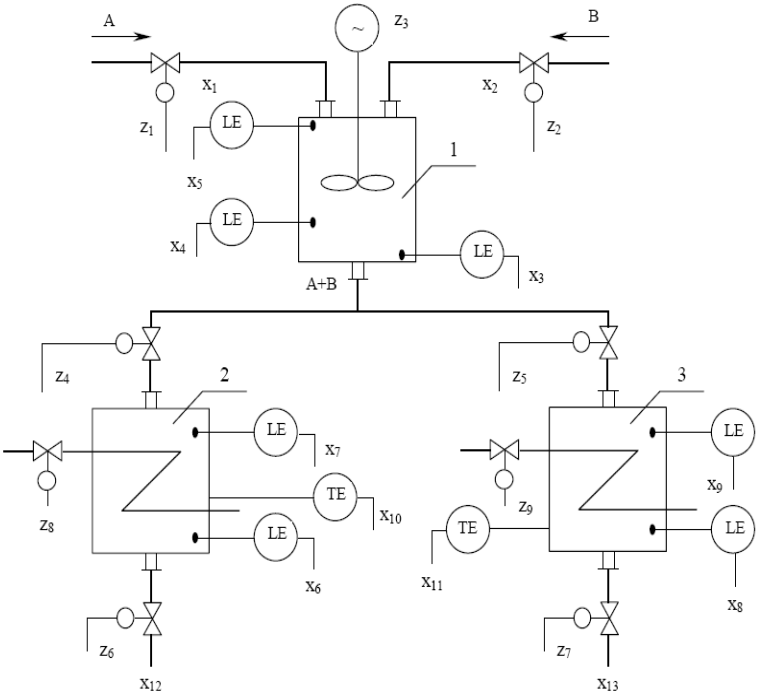


Рис. 6. Технологическая установка для приготовления раствора

Таблица 2

Входные сигналы управляющего автомата

Сигнал	Описание
x1	Подача команды на открытие клапана z1 (налив компонента А)
x2	Подача команды на открытие клапана z2 (налив компонента В)
x3	Аппарат 1 пуст
x4	В аппарат 1 налито 20 л компонента А
x5	В аппарат 2 налито 80 л компонента В
x6	Аппарат 2 пуст
x7	В аппарат 2 налито 100 л смеси (А+В)
x8	Аппарат 3 пуст
x9	В аппарат 3 налито 100 л смеси (А+В)
x10	В аппарате 2 температура достигла 50 °С
x11	В аппарате 3 температура достигла 80 °С

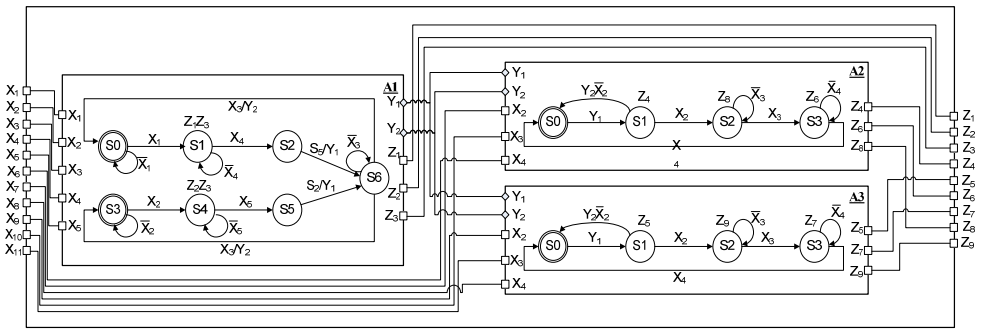
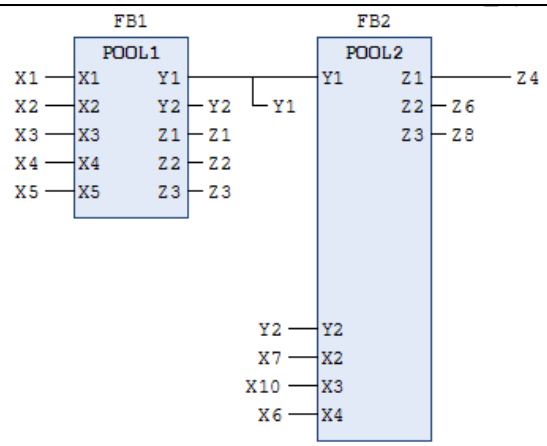
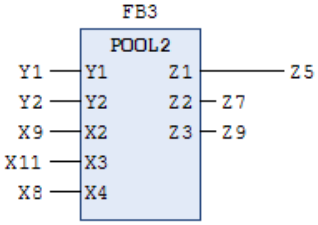


Рис. 7. ИМНДА для управления технологической установкой

Таблица 3

Реализация составного модуля на языке *FBD*

Цепь Q_1	
Цепь Q_2	

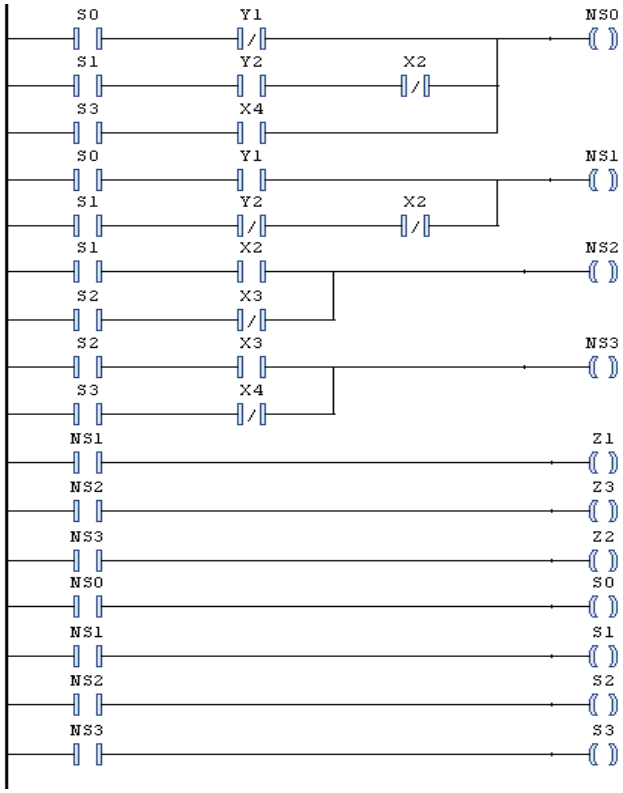


Рис. 8. Реализация модуля А2 на языке *LD* в *CoDeSys*

Заключение

В работе предложен и проиллюстрирован подход к реализации ПЛК-ориентированных систем управления для ДСС с использованием модели ИМНДА. Рассмотрена структура инструментальных средств поддержки методологии проектирования и реализации систем управления на основе ИМНДА. Направлением дальнейших исследований является развитие и усовершенствование инструментальных средств, а также их использование в конкретных предметных областях.

Список литературы

1. **Cassandras, C. G.** Introduction to Discrete Event Systems / C. G. Cassandras, S. Lafortune. – Springer, 2008. – 772 p.
2. **Петров, И. В.** Программируемые контроллеры. Стандартные языки и приемы прикладного программирования / И. В. Петров ; под ред. В. П. Дьяконова. – М. : СОЛОН-Пресс, 2004. – 256 с.
3. International Standard IEC 61131-3 (edition 2.0) : Programmable Controllers / International Electrotechnical Commission. – Geneva, 2003. – 230 p.
4. **Frey, G.** Formal methods in PLC programming / L. Litz // Proc. IEEE Conference on Systems, Man, and Cybernetics (SMC2000). – Nashville, USA, 2000, Oct. – P. 2431–2436.
5. **Шалыто, А. А.** SWITCH-технология. Алгоритмизация и программирование задач логического управления / А. А. Шалыто. – СПб. : Наука, 1998. – 628 с.
6. **Dierks, H.** PLC-Automata: A New Class of Implementable Real-Time Automata / H. Dierks // Transformation-Based Reactive Systems Development (ARTS‘97). – Springer LNCS, 1997. – Vol. 1231. – P. 111–125.
7. **Shah, S. S.** Reconfigurable logic control using modular FSMs: Design, verification, implementation, and integrated error handling / S. S. Shah, E. W. Endsley, M. R. Lucas, D. M. Tilbury // Proceedings of the 2002 American Control Conference. – 2002. – Vol. 5. – pp. 4153–4158.
8. **Thapa, D.** Modeling, Verification, and Implementation of PLC program using Timed-MPSG / D. Thapa, S. C. Park, C. M. Park, G.-N. Wang // Proceedings of the 2007 Summer Computer Simulation Conference (SCSC'07). – San Diego, CA, USA, 2007. – P. 533–540.
9. **Sacha, K.** Translatable finite state time machine / K. Sacha // Lecture Notes in Computer Science. – 2007. – Vol. 4745. – P. 117–132.
10. **Moura, R. S.** Using basic Statechart to program industrial controllers / R. S. Moura, L. A. Guedes // Computer Standards & Interfaces. – 2012. – № 34. – P. 60–67.
11. **Frey, G.** A Re-Engineering Approach for PLC Programs using Finite Automata and UML / G. Frey, M. B. Younis // Proc. IEEE International Conference on Information Reuse and Integration, IRI-2004. – Las Vegas, USA, 2004, Nov. – P. 24–29.
12. **Дубинин, В. Н.** Проектирование и реализация систем управления дискретными событийными системами на основе иерархических модульных недетермирированных автоматов (Часть 1. Формальная модель) / В. Н. Дубинин, Д. Н. Дроздов, Д. А. Будаговский // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2016. – № 1 (37). – С. 30–41.
13. **Вашкевич, Н. П.** Недетерминированные автоматы и их использование для синтеза систем управления / Н. П. Вашкевич, С. Н. Вашкевич. – Пенза : Изд-во Пенз. гос. ун-та, 1996. – 88 с.
14. **Fantuzzi, C.** A Design Pattern for translating UML software models into IEC 61131-3 Programming Languages / C. Fantuzzi, M. Bonfe, F. Fanfoni, C. Secchi // 18th

IFAC World Congress (August 28 – September 2, 2011). – Milano (Italy), 2011. – Vol. 18, Part 1. – P. 9158–9163.
15. Simulink PLC Coder. – URL: http://www.mathworks.com/products/sl-plc-coder/.
16. ViVe – VisualVerifier Tool Framework. – URL: http://www.fb61499.com/license.html.

References

1. Cassandras C. G., Lafortune S. *Introduction to Discrete Event Systems*. Springer, 2008, 772 p.
2. Petrov I. V. *Programmiruemye kontrollery. Standartnye yazyki i priemy prikladnogo programmirovaniya* [Programmable controllers. Standard languages and methods of applied programming]. Moscow: SOLON-Press, 2004, 256 p.
3. *International Standard IEC 61131-3 (edition 2.0): Programmable Controllers*. International Electrotechnical Commission. Geneva, 2003, 230 p.
4. Frey G., Litz L. *Proc. IEEE Conference on Systems, Man, and Cybernetics (SMC2000)*. Nashville, USA, 2000, Oct., pp. 2431–2436.
5. Shalyto A. A. *SWITCH-tekhnologiya. Algoritmizatsiya i programmirovanie zadach logicheskogo upravleniya* [SWITCH technology. Algorithmization and programming of logic control problems]. Saint-Petersburg: Nauka, 1998, 628 p.
6. Dierks H. *Transformation-Based Reactive Systems Development (ARTS'97)*. Springer LNCS, 1997, vol. 1231, pp. 111–125.
7. Shah S. S., Endsley E. W., Lucas M. R., Tilbury D. M. *Proceedings of the 2002 American Control Conference*. 2002, vol. 5, pp. 4153–4158.
8. Thapa D., Park S.C., Park C. M., Wang G.-N. *Proceedings of the 2007 Summer Computer Simulation Conference (SCSC'07)*. San Diego, CA, USA, 2007, pp. 533–540.
9. Sacha K. *Lecture Notes in Computer Science*. 2007, vol. 4745, pp. 117–132.
10. Moura R. S., Guedes L. A. *Computer Standards & Interfaces*. 2012, no. 34, pp. 60–67.
11. Frey G. A, Younis M. B. *Proc. IEEE International Conference on Information Reuse and Integration, IRI-2004*. Las Vegas, USA, 2004, Nov., pp. 24–29.
12. Dubinin V. N., Drozdov D. N., Budagovskiy D. A. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2016, no. 1 (37), pp. 30–41.
13. Vashkevich N. P., Vashkevich S. N. *Nedeterminirovannye avtomaty i ikh ispol'zovanie dlya sinteza sistem upravleniya* [Non-deterministic automata and their application in control system sunthesis]. Penza: Izd-vo Penz. gos. un-ta, 1996, 88 p.
14. Fantuzzi C. A, Bonfe M., Fanfoni F., Secchi C. *18th IFAC World Congress (August 28 – September 2, 2011)*. Milano (Italy), 2011, vol. 18, part 1, pp. 9158–9163.
15. *Simulink PLC Coder*. Available at: http://www.mathworks.com/products/sl-plc-coder/.
16. *ViVe – VisualVerifier Tool Framework*. Available at: http://www.fb61499.com/license.html.

Дубинин Виктор Николаевич
доктор технических наук, профессор, кафедра вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: dubinin.victor@gmail.com

Dubinin Viktor Nikolaevich
Doctor of engineering sciences, professor, sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Будаговский Дмитрий Александрович
аспирант, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: budagovsky92@gmail.com

Budagovskiy Dmitriy Aleksandrovich
Postgraduate student, Penza State University (40 Krasnaya street, Penza, Russia)

Дроздов Дмитрий Николаевич
аспирант, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: dmitriidrozdov9@gmail.com

Drozdov Dmitriy Nikolaevich
Postgraduate student, Penza State University (40 Krasnaya street, Penza, Russia)

Артамонов Дмитрий Владимирович
доктор технических наук, профессор, кафедра автономных информационных и управляющих систем, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: aius@pnzgu.ru

Artamonov Dmitriy Vladimirovich
Doctor of engineering sciences, professor, sub-department of autonomous information and control systems, Penza State University (40 Krasnaya street, Penza, Russia)

УДК 681.5

Дубинин, В. Н.

Проектирование и реализация систем управления дискретными событийными системами на основе иерархических модульных недетерминированных автоматов (Ч. 2. Методы и средства) / В. Н. Дубинин, Д. А. Будаговский, Д. Н. Дроздов, Д. В. Артамонов // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2016. – № 2 (38). – С. 18–32. DOI 10.21685/2072-3059-2016-2-2