УДК 681.5
DOI 10.21685/2072-3059–2017-1-2

В. Н. Дубинин, Л. П. Климкина, А. В. Калачев

МНОГОУРОВНЕВЫЕ СИСТЕМЫ ФУНКЦИОНАЛЬНЫХ БЛОКОВ IEC 61499 И ИХ ПРЕОБРАЗОВАНИЕ К ОДНОУРОВНЕВЫМ МОДЕЛЯМ

Аннотация.

Актуальность и цели. В настоящее время одна из ключевых технологий в построении систем управления в промышленной автоматизации опирается на стандарт IEC 61499. В общем случае системы управления на основе функциональных блоков (ФБ) стандарта IEC 61499 представляют собой сложные иерархические системы, включающие множество артефактов и связей. Процесс проектирования систем данного класса предполагает разработку спектра взаимосвязанных моделей и их взаимных преобразований. Цель данного исследования – разработка методики перехода от многоуровневых систем функциональных блоков к одноуровневым графовым моделям.

Материалы и методы. Данное исследование проведено с использованием теории множеств, теории конечных автоматов, теории графов и графовых трансформаций.

Результаты. Разработаны формальные основы иерархической организации систем управления, построенных с использованием стандарта IEC 61499, включая определение системной конфигурации и ее развертывания, принципы буферирования данных в межмодульных и межуровневых интерфейсах, что позволяет перейти от иерархических структур систем ФБ к одноуровневым моделям, которые можно принять за каноническую форму представления систем ФБ. Предложен подход к формализации процесса перехода от многоуровневых систем ФБ к одноуровневым на основе трансформации графов. В рамках формализации этого процесса разработаны: набор метамоделей систем ФБ в виде типизированных атрибутных графов; набор правил трансформации для перехода от многоуровневой структуры систем ФБ к одноуровневой структуре.

Выводы. Предложенный метод перехода от многоуровневых систем функциональных блоков к одноуровневым графовым моделям сокращает число межуровневых связей и повышает однородность модели, что, в свою очередь, будет способствовать повышению эффективности анализа систем управления на стадии верификации и имитационного моделирования.

Ключевые слова: функциональный блок, стандарт IEC 61499, системы управления, иерархическая структура, развертывание, метамодель, графовая модель, трансформация.

V. N. Dubinin, L. P. Klimkina, A. V. Kalachev

MULTILAYER SYSTEMS OF IEC 61499 FUNCTION BLOCKS AND THEIR TRANSFORMATION TO ONE-LEVEL MODELS

Abstract.

Background. Currently, one of the key technologies in the design of control systems for industrial automation relies on the standard IEC 61499. In general, the control systems based on IEC 61499 function blocks are complex hierarchical systems

including a plurality of artifacts and relations. The process of designing the systems of this class involves the development of a set of interconnected models and their mutual transformations. The goal of this work is to development a method of unfolding and transition from multilayer function block systems to one-level graph models.

Materials and methods. This research was conducted using the set theory, the automata theory, the graph theory and the theory of graph transformations.

Results. In this work we have developed formal foundations of the hierarchical organization of control systems based on the standard IEC 61499, including the definition of a system configuration and its unfolding, the principles of data buffering in the inter-module and inter-level interfaces that allows one to come from hierarchical structures of function block systems to one-level models that can be taken as a canonical form of representation of function block systems. An approach to formalization of the process of transition from multilayer systems to the one-level models on the basis of graph transformation is proposed. As part of the formalization process the following have been developed: 1) a set of metamodels of function block systems in the form of typed attributed graphs; 2) a set of transformation rules for the transition from the multilayer structure to the one-level one.

Conclusions. The proposed method of transition from the multilayer function block systems to one-level graph models reduces the number of inter-level connections and improves the homogeneity of the model that in turn will enhance the efficiency of the analysis of control systems at the stage of verification and simulation.

Key words: function block, standard IEC 61499, control system, hierarchical structure, unfolding, metamodel, graph model, transformation.

Введение

С учетом существующих тенденций и достижений в сфере новых технологий разработки программного обеспечения (ПО) и телекоммуникаций в 2005 г. Международной электротехнической комиссией был принят новый стандарт IEC 61499 [1]. Этот стандарт определяет путь для построения систем управления промышленными процессами нового поколения [2]. Это интеллектуальные распределенные компонентно-базированные системы. Архитектура IEC 61499 строится на основе определений IEC 61131-3. Основной элемент архитектуры – функциональный блок (ФБ), хорошо знакомый и широко используемый инженерами, но его концепция расширена в нескольких направлениях для того, чтобы отразить новые достижения в области проектирования современных программных систем. Одно из основных расширений ФБ – *событийный интерфейс*, который позволяет явно определить последовательности выполнения ФБ. Другое расширение ФБ связано с объектной и компонентной ориентацией.

Большое значение для разработки систем автоматизированного проектирования управляющих систем на основе ФБ имеет формализация как основных артефактов проектирования IEC 61499, так и процесса их преобразования для проведения основных фаз проектирования: спецификации, верификации, оценки производительности и реализации. Важным преобразованием является переход от многоуровневого представления системы к одноуровневому. Использование одноуровневых моделей, как правило, повышает эффективность верификации и имитационного моделирования.

Перспективным подходом к трансформации моделей является подход, основанный на трансформации графов [3]. Данный подход развивается и уже нашел применение в технологии проектирования программного обеспечения на основе управления моделями (MDE) [4]. Например, он был успешно применен для рефакторинга диаграмм ECC в базисных ФБ [5]. Преимуществом подхода является его формальность, что позволяет избежать многих ошибок проектирования и проверить правила на корректность. Этот подход был принят в данной работе за основу. В качестве метамоделей при этом используются типизированные атрибутные графы (ТАГ) [3]. ТАГ и трансформация на их основе поддерживаются системой AGG, которая является одним из наиболее популярных средств для трансформаций графов [6].

1. Краткий обзор стандарта IEC 61499

Управляющая система представляется совокупностью устройств, взаимодействующих друг с другом с помощью коммуникационной сети, состоящей из сегментов и линий связи. Функция, выполняемая системой управления, описывается с использованием приложения.

Устройство представляет собой физическую сущность, способную выполнять одну или несколько специфицированных функций в определенном контексте и имеющую, по меньшей мере, один интерфейс, являющийся или интерфейсом управляемого процесса, или коммуникационным интерфейсом [1]. На практике в качестве устройства могут выступать программируемые логические контроллеры (ПЛК), программируемые контроллеры автоматизации (ПКА), промышленные компьютеры. Устройство состоит из одного или нескольких ресурсов.

Ресурс – это функциональная единица, которая имеет независимое управление своими операциями, включая планирование и выполнение алгоритмов [1]. Важнейшей функцией ресурса является функция планирования (диспетчирования), которая определяет порядок выполнения ФБ и передачу данных между ними.

Приложение является программной функциональной единицей, предназначенной для решения определенной задачи в системе управления. Приложение представляется в виде сети связанных между собой ФБ и субприложений, которые могут выполняться на различных ресурсах и устройствах системы. Модель приложения представлена на рис. 1.

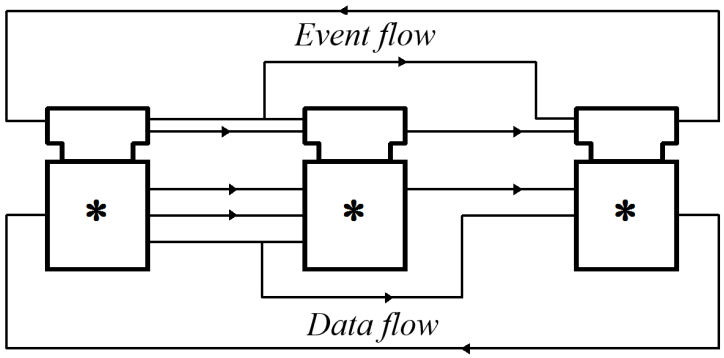


Рис. 1. Модель приложения IEC 61499

На рис. 2 приведен пример соотношения приложений, устройств, ресурсов и сегментов в архитектуре IEC 61499.

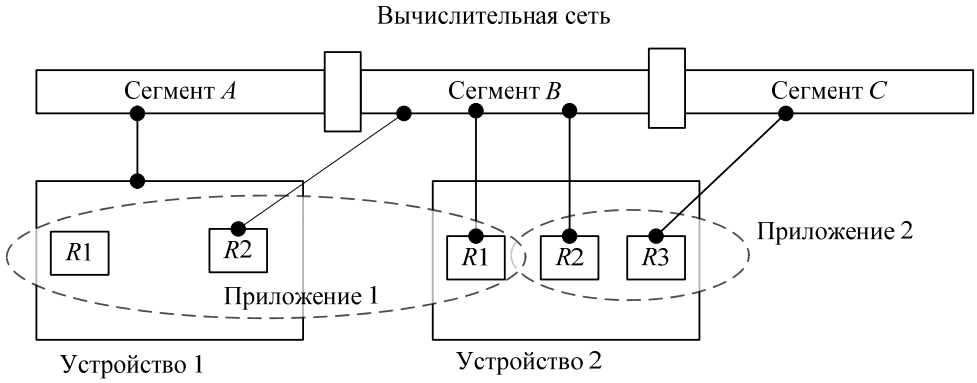
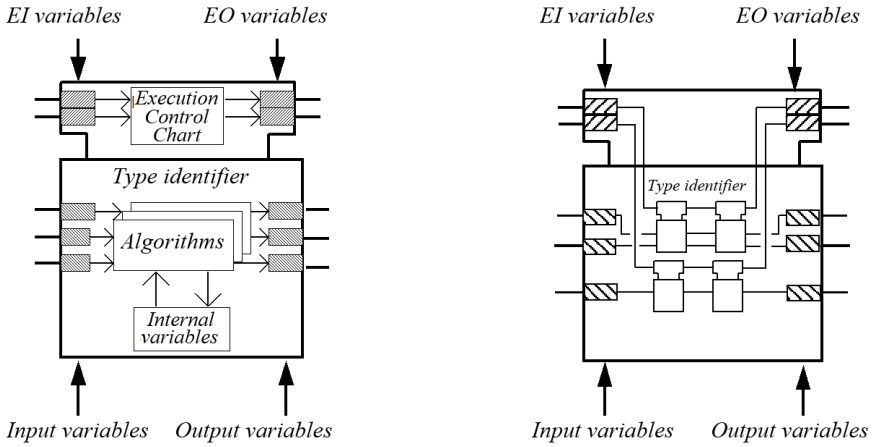


Рис. 2. Иллюстрация системных конфигураций

Основным артефактом проектирования в IEC 61499 является функциональный блок, представляющий собой специфичный программный компонент, имеющий интерфейс. ФБ представляет собой независимую программную единицу, которая может быть реализована, протестирована и использована отдельно от других ФБ. Интерфейс специфицируется множеством событийных и информационных входов и выходов, через которые соответствующий элемент взаимодействует с окружением.

Существует три вида ФБ: базисный ФБ, составной ФБ и сервисный интерфейсный ФБ (СИФБ). Структуры первых двух типов ФБ приведены на рис. 3. Функциональность базисного ФБ определяется машиной состояний (*Execution Control Chart – ECC*) типа автомата Мура. Базисный ФБ может содержать несколько алгоритмов. Эти алгоритмы связаны с состояниями ECC, они являются невидимыми извне и к ним нет прямого доступа.



а) ***б***)

Рис. 3. Функциональные блоки стандарта IEC 61499:
а – базисный ФБ; ***б*** – составной ФБ

Функциональность составного ФБ определяется входящей в него сетью ФБ. Сервисные интерфейсные функциональные блоки предоставляют один или несколько сервисов приложению на основе отображения сервисных примитивов на входы и выходы ФБ. СИФБ работают как «обертка», абстрагируя лежащие ниже аппаратные средства.

2. Модель системной конфигурации

Рассмотрим формально основные артефакты проектирования, определенные в стандарте IEC 61499.

Конфигурация системы s (далее – просто система s) определяется кортежем:

$$s = (D, DT, dtype, RT, FBT, SAT, A, fbmap, SG, rmap),$$

где $D = \{d_1, d_2, ..., d_{N_D}\}$ – конечное непустое множество конфигураций устройств (далее – просто устройств), входящих в систему s; $DT = \{dt_1, dt_2, ..., dt_{N_{DT}}\}$ – конечное, возможно пустое, множество типов устройств, используемых в системе s; $dtype$: $D \to DT \cup \{none\}$ – функция, назначающая устройствам тип устройств; $RT = \{rt_1, rt_2, ..., rt_{N_{RT}}\}$ – конечное, возможно пустое, множество типов ресурсов, используемых в системе s; $FBT = \{fbt_1, fbt_2, ..., fbt_{N_{FBT}}\}$ – конечное непустое множество типов ФБ, используемых в системе s; $SAT = \{sat_1, sat_2, ..., sat_{N_{SAT}}\}$ – конечное, возможно пустое, множество типов субприложений, используемых в системе s; $A = \{a_1, a_2, ..., a_{N_A}\}$ – конечное, возможно пустое, множество приложений системы S; $fbmap$: $FB^A \to FB^S$ – отображение ФБ приложения на ФБ устройств и ресурсов. Здесь под FB^A понимаются все ФБ, входящие в состав приложения и всех (рекурсивно) вложенных в него субприложений; SG – множество сегментов локальной вычислительной сети (ЛВС), на которых выполняются приложения; $rmap$: $RS \to SG$ – функция распределения ресурсов системы s по сегментам ЛВС. Если все ресурсы некоторого устройства относятся к одному и тому же сегменту, то считается, что все устройство в целом относится к этому сегменту.

Устройство $d_i \in D$ определяется следующим образом:

$$d_i = \left(R^{d_i}, RT, rtype^{d_i}, FBN^{d_i} \right),$$

где $R^{d_i} = \left\{ r_1^{d_i}, r_2^{d_i}, ..., r_{N_{R^{d_i}}}^{d_i} \right\}$ – конечное непустое множество ресурсов, входящих в состав устройства $d_i \in D$; $rtype^{d_i} : R^{d_i} \to RT \cup \{none\}$ – функция, ставящая в соответствие ресурсам устройства $d_i \in D$ тип ресурса; FBN^{d_i} – сеть ФБ, размещенная на устройстве $d_i \in D$.

Тип устройства $dt_j \in DT$ определяется практически так же, как и устройство:

$$dt_j = \left(R^{dt_j}, RT, rtype^{dt_j}, FBN^{dt_j}, Par^{dt_j} \right).$$

В данном случае дополнительный компонент кортежа Par^{dt_j} определяет множество параметров типа устройства $dt_j \in DT$. Каждый параметр есть некоторая константа определенного типа, подаваемая на какой-либо информационный вход сети ФБ FBN^{dt_j}.

Ресурс $r_i \in R$ определяется только своей сетью ФБ $r_i = (FBN^{r_i})$.

Тип ресурса $rt_j \in RT$ задается двойкой $rt_j = (FBN^{rt_j}, Par^{rt_j})$, где Par^{rt_j} – множество параметров типа ресурса $rt_j \in RT$.

Сеть ФБ *FBN* в общем виде определяется кортежем:

$$FBN = (FB, fbtype, FBT, EvConn, DataConn),$$

где *FB* – непустое множество компонентных ФБ сети *FBN*; *fbtype*: $FB \rightarrow FBT$ – функция, назначающая компонентным ФБ тип ФБ; *EvConn* – непустое множество событийных связей сети ФБ *FBN*; *DataConn* – непустое множество информационных связей сети ФБ *FBN*.

Все множество типов ФБ делится на три класса (сорта):

$$FBT = BFBT \cup CFBT \cup SIFBT; \quad BFBT \cap CFBT \cap SIFBT = \varnothing,$$

где *BFBT*, *CFBT* и *SIFBT* – множество типов базисных, составных и сервисных интерфейсных ФБ соответственно.

Формальное определение типа ФБ является специфичным для каждого сорта и будет приведено в последующих подразделах.

3. Развертывание системной конфигурации

Изначально в соответствии со стандартом IEC 61499 система управления представляется в «свернутом» виде, скорее, на уровне типов, чем на уровне экземпляров. Для имитационного моделирования системы управления, ее исследования, а также интерпретации в режиме реального времени (для управления реальными объектами) требуется представление системы на уровне экземпляров. Назовем процесс перехода от системы типов к системе экземпляров в IEC 61499 *развертыванием* системной конфигурации. Развертывание системной конфигурации включает:

1) развертывание ресурсов и устройств;

2) развертывание ФБ-подобных элементов, под которыми будем понимать составные ФБ, субприложения, приложения, а также сети ФБ, используемые на ресурсах и устройствах.

Следует отметить, что эти два типа развертываний существенно различаются.

Общий алгоритм развертывания системной конфигурации имеет вид:

procedure *unfoldSys*(*s*)

  do forall $d_i \in D$

    $dt_j = dtype(d_i)$

    $R_{ext}^{d_i} = R^{d_i} \cup R^{dt_j}$

    $FBN_{ext}^{d_i} = FBN^{d_i} \cup FBN^{dt_j}$

    do forall $r_k \in R_{ext}^{d_i}$

rt_m=*rtype*(r_k)

$FBN_{ext}^{r_k} = FBN^{r_k} \cup FBN^{rt_m}$

unfoldFB($FBN_{ext}^{r_k}$)

end_forall

end_forall

end_procedure

Здесь объекты с нижним индексом *ext* есть исходные объекты (первоначально – объекты без этого индекса), расширенные в результате развертывания. В соответствии с приведенным алгоритмом развертывание устройства сводится к «миграции» всех ресурсов из типа устройства в множество ресурсов собственно устройства, а также объединению сетей ФБ, определенных в устройстве и соответствующем ему типе устройства. Развертывание ресурса включает только добавление сети ФБ, принадлежащей типу ресурса, к сети ФБ, расположенной на самом ресурсе. Дальнейший этап развертывания включает развертывание всех сетей ФБ, расположенных на ресурсах, а также развертывание приложений (если таковые имеются).

Рассмотрим подробнее развертывание ФБ-подобных объектов. Назовем *ссылочным экземпляром* ФБ (субприложения) компонентный ФБ (субприложение), используемый в процессе развертывания. *Развернутым экземпляром* ФБ или субприложения будем называть содержимое (контент), который соответствует ссылочному экземпляру ФБ или субприложения. Развертывание ФБ-подобного объекта сводится к рекурсивной замене ссылочных экземпляров объектов на соответствующие им развернутые экземпляры. Они получаются путем клонирования описания типа, соответствующего ссылочному объекту. Синтаксически развернутый экземпляр ФБ является копией соответствующего типа. Поэтому в дальнейшем при их описании будем пользоваться введенными обозначениями из описания соответствующих типов. Для определенности речь ниже будет идти о развертывании составных ФБ, однако те же самые рассуждения являются действительными и для всех ФБ-подобных элементов. Сеть ФБ на ресурсе или устройстве можно представить в виде составного ФБ без входов и выходов.

4. Переход от иерархических структур систем функциональных блоков к одноуровневым структурам

Система экземпляров ФБ полностью определяется деревом иерархии (развернутых) экземпляров ФБ, которое можно определить следующим кортежем:

$$FTree = (M, Aggr, fbtype', idM),$$

где $M = \{M_1, M_2, ..., M_{N_M}\}$ – множество развернутых экземпляров ФБ; $Aggr \subseteq M \times M$ – отношение агрегации; $fbtype'$: $M \rightarrow FBT$ – функция разметки экземпляров ФБ типами ФБ; idM: $M \rightarrow ID^M$ – функция назначения экземплярам ФБ уникальных идентификаторов.

Алгоритм построения дерева иерархии экземпляров рассмотрен в [7].

Замена ссылочного экземпляра развернутым экземпляром производится в три этапа:

1) добавление развернутого экземпляра;
2) встраивание развернутого экземпляра;
3) удаление ссылочного экземпляра.

Встраивание развернутого экземпляра производится путем перекоммутации событийных и информационных линий с входов-выходов ссылочного экземпляра на соответствующие входы-выходы развернутого экземпляра. Между интерфейсами развернутого и ссылочного экземпляров должно существовать взаимно однозначное соответствие. На рис. 4 продемонстрировано развертывание развернутого экземпляра M_i. Он был получен из ссылочного экземпляра fb_i. В процессе развертывания M_i ссылочные экземпляры $fb_{i+1}...fb_{i+n}$ заменяются на соответствующие им развернутые экземпляры $M_{i+1}...M_{i+n}$.

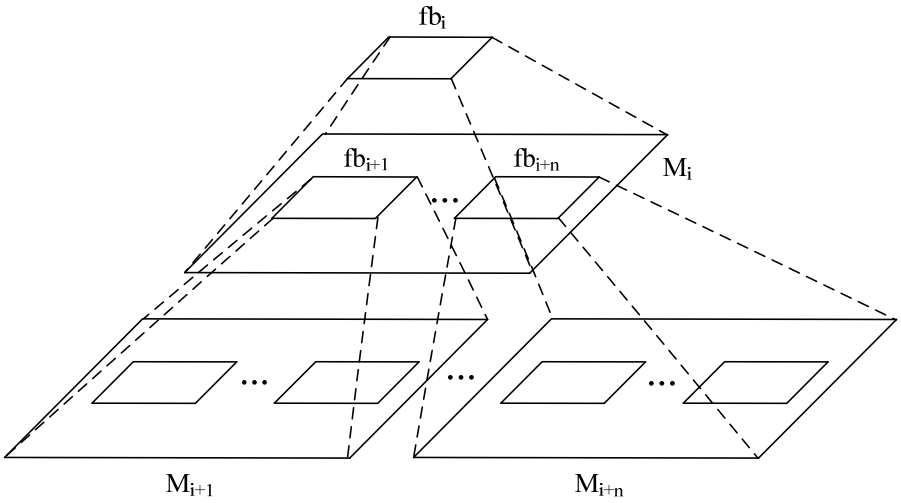


Рис. 4. Развертывание экземпляра ФБ

5. Буферирование данных

Очевидно, что при интерпретации систем ФБ необходимо учитывать семантику синтаксических конструкций. Рассмотрим более подробно передачу данных между (интерфейсами) ФБ (рис. 5).

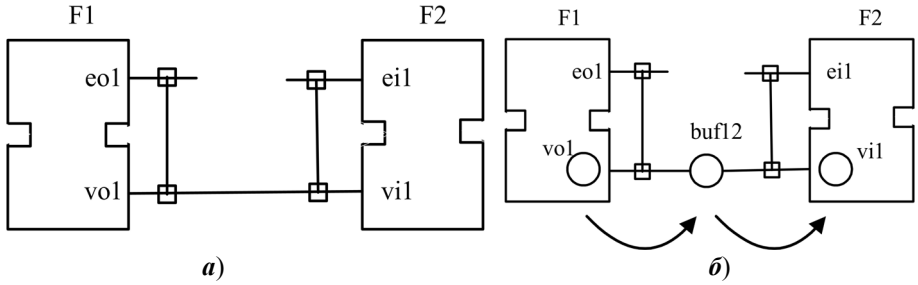


Рис. 5. Передача данных между ФБ с буферизацией: синтаксическое представление (***а***) и семантическое представление (***б***)

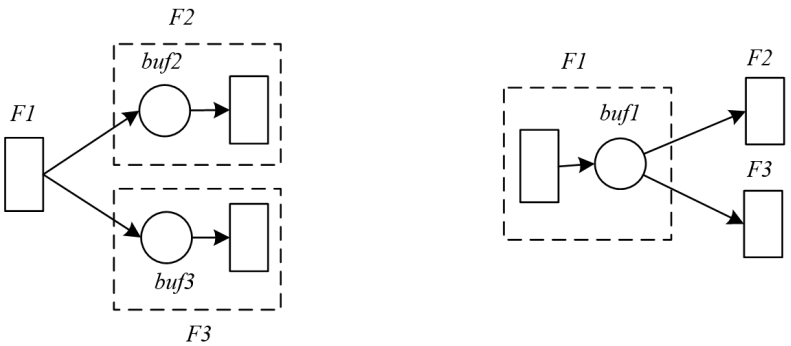
В соответствии с первой частью стандарта IEC 61499 [1] каждому информационному входу и выходу ФБ ставится в соответствие переменная,

в которой хранится текущее значение входов и выходов. Информационная связь между выходом одного ФБ и входом другого ФБ, по сути, определяет передачу данных между соответствующими переменными. Однако передача данных между парой ФБ не представляет собой одномоментный (неделимый) акт вследствие того, что выдача и съем данных могут инициироваться разными сигналами. Поэтому в тракт передачи данных необходимо включить буфер (рис. 5,***б***). На рис. 5,***б*** при выдаче сигнала eo1 блоком F1 данные из выходной переменной vo1 переписываются в буфер buf12, а при съеме данных в результате обработки сигнала ei1 в блоке F2 данные из буфера buf12 передаются во входную переменную vi1. Поскольку буфера данных не определены в стандарте IEC 61499 как самостоятельные элементы (точнее, в стандарте IEC 61499 вообще нет ссылок на них), но в то же время являются необходимыми элементами, то встает вопрос об их размещении. Единственная возможность решения этой проблемы – отнести их к ФБ. Существует два структурных варианта «встраивания» буферов данных в ФБ:

1) соотнести буфер данных с информационным входом ФБ;

2) соотнести буфер данных с информационным выходом ФБ.

Недостатком первого варианта является необходимость дублирования данных по всем исходящим из некоторого выхода информационным связям (рис. 6,***а***). Учитывая топологические ограничения на структуру взаимосвязей ФБ в стандарте IEC 61499 [1], согласно которым к информационному входу ФБ может быть подключено не более одной информационной линии, наиболее экономичен и естественен второй вариант (рис. 6,***б***).



а) ***б***)

Рис. 6. Встраивание буферов данных в ФБ: ***а*** – вариант встраивания «один буфер на один вход ФБ»; ***б*** – вариант встраивания «один буфер на один выход ФБ»

Как видно из рис. 4, система ФБ представляет собой иерархическую многоуровневую структуру. Иерархичность системы возникает из-за использования составных ФБ и субприложений. Наличие межуровневых связей зачастую усложняет процесс моделирования и интерпретации систем ФБ. Для избавления от фактора иерархичности предлагается одноуровневое представление (*flat*) систем ФБ, эквивалентное исходной системе. Иногда такое представление называют «*плоским*». Одноуровневое представление могло бы также служить неким каноническим представлением сложных многоуровневых систем ФБ.

Однако в случае систем ФБ переход к одноуровневому представлению (*flattening*) сопряжен с определенными трудностями. Это связано с тем, что входной и выходной интерфейсы ФБ имеют определенную логику съема и выдачи данных, что связано с наличием в интерфейсах *WITH*-связей. Для решения этой проблемы предлагается отделить интерфейсную логику ФБ от основной функции, выполняемой ФБ. Для реализации входной и выходной логики ФБ было введено понятие *клапана данных* (КД) [7], реализующего передачу данных через интерфейсы.

Сети КД используются для моделирования систем интерфейсов ФБ и их представления в одноуровневой структуре ФБ (рис. 7). Если КД представляет входной интерфейс ФБ, то назовем его входным, а если выходной – то выходным.

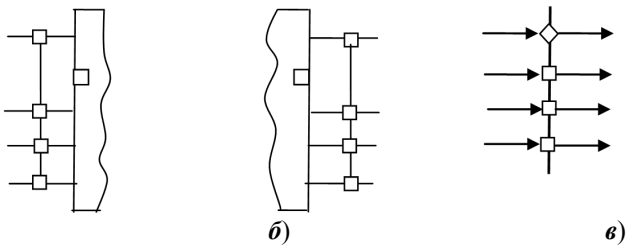


Рис. 7. Представление входного (***а***) и выходного (***б***) интерфейсов составного ФБ с помощью клапана данных (***в***)

В одноуровневом представлении системы ФБ могут присутствовать базисные ФБ, СИФБ, а также КД. Составные ФБ и субприложения в результирующей системе отсутствуют.

6. Графовая метамодель систем функциональных блоков

Ниже определяется визуальный язык функциональных блоков, основанный на их представлении в виде типизированных атрибутных графов (ТАГ) [3]. Данный визуальный язык основан на стандарте *IEC* 61499 [1]. Мотивом разработки языка является создание языковой базы для трансформации описаний систем ФБ в другие модельные формы. Кроме «метамодельного» определения начального языка ФБ, в данном разделе представлены промежуточные метамодели, определяющие системы ФБ разных уровней, используемые в процессе синтеза формальных моделей.

Произведем классификацию представлений систем ФБ на основе степени полноты генерации одноуровневого представления. Будем выделять системы уровней 0, 1 и 2. Системами ФБ *уровня 0* являются системы, в которых не раскрыт ни один (ссылочный экземпляр) ФБ. Система ФБ *уровня 0* представляет исходную многоуровневую систему ФБ в виде иерархической типизированной структуры в начальной форме. В системах *уровня 1* раскрыты все составные ФБ, базисные ФБ не раскрываются. Система ФБ *уровня 2* представляет одноуровневую систему ФБ, в которой раскрыты все блоки, включая базисные. Эти системы состоят только из «начинок» базисных ФБ, связанных сетью клапанов данных. Эту форму представления системы ФБ можно считать нормализованной, поскольку ее дальнейшее раскрытие невозможно. Системы *уровня 1* можно считать промежуточной формой.

К введенному выше признаку классификации систем ФБ добавим признак, определяющий, является ли система ФБ замкнутой или разомкнутой. Будем считать, что замкнутая система ФБ не имеет входов и выходов.

На рис. 8 представлена метамодель базисного ФБ в виде типизированного графа. В данной метамодели используются следующие типы вершин: *CEI* и *CEO* – событийные вход и выход оболочки ФБ соответственно; *CDI* и *CDO* – информационные вход и выход оболочки ФБ соответственно; *S* – *EC*-состояние; *Cond* – условие *EC*-перехода; *Act* – *EC*-акция; *Var* – внутренняя переменная.

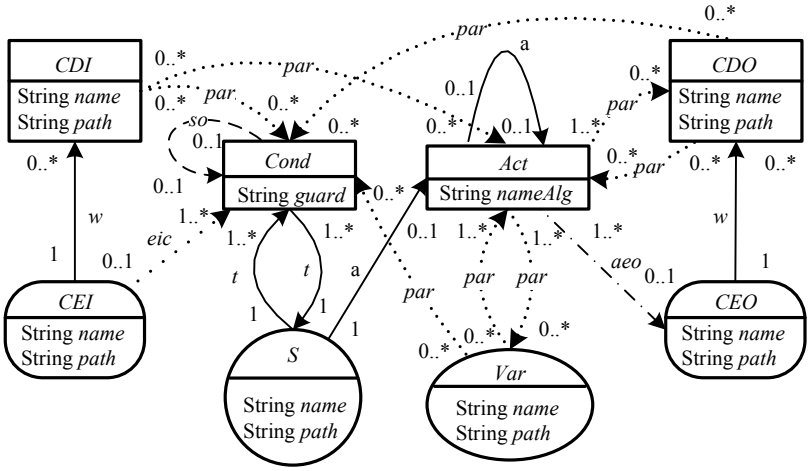


Рис. 8. Метамодель базисного ФБ

В метамодели базисного ФБ используются следующие типы дуг: *w* – *WITH*-связь; *eic* – дуга, связывающая событийный вход с условием *EC*-перехода; *aeo* – дуга, связывающая *EC*-акцию с событийным выходом; *t* – дуга, связывающая *EC*-состояния и *EC*-условия; *a* – дуга, связывающая *EC*-состояние с первой *EC*-акцией или *EC*-акции между собой согласно порядку их выполнения; *so* – дуга, определяющая порядок оценки *EC*-переходов, выходящих их одного *EC*-состояния; *par* – дуга для представления потоков данных через алгоритмы. Данная дуга служит также для определения параметров условия *EC*-перехода.

На рис. 9 приведена метамодель составного ФБ (и субприложения) в виде типизированного графа. Штрих-пунктирные линии для связей типов *ec* и *dc* относятся только к субприложению, а связи типа *w* относятся только к составному ФБ. Как видно из рис. 9, метамодель составного ФБ существенно отличается от метамодели базисного ФБ. Общими являются лишь типы вершин, определяющие интерфейс ФБ (вершины типов *CEI*, *CEO*, *CDI*, *CDO*). Вершина типа *FB* представляет ссылочный экземпляр ФБ (субприложения), а вершины типов *EI*, *EO*, *DI*, *DO* – его интерфейс, причем вершина типа *EI* представляет событийный вход, вершина типа *EO* – событийный выход, вершина типа *DI* – информационный вход, а вершина типа *DO* – информационный выход. Дуги типа *b* определяют принадлежность данных интерфейсных вершин определенному ссылочному экземпляру ФБ (субприложения). Для представления потока событий используются дуги типа *ec*, а для представления потока данных – дуги типа *dc*.

Метамодель замкнутых систем ФБ *уровня 2* в виде типизированного графа представлена на рис. 10. Данная метамодель состоит из элементов, используемых в других метамоделях. Следует также отметить, что в процессе генерации одноуровневого представления системы ФБ, кроме отмеченных выше элементов, используются следующие виды вершин: вершина *NumM* (представляет счетчик модулей) и вершина *ParentType* (используется для хранения имени типа ФБ, порождающего экземпляр ФБ).

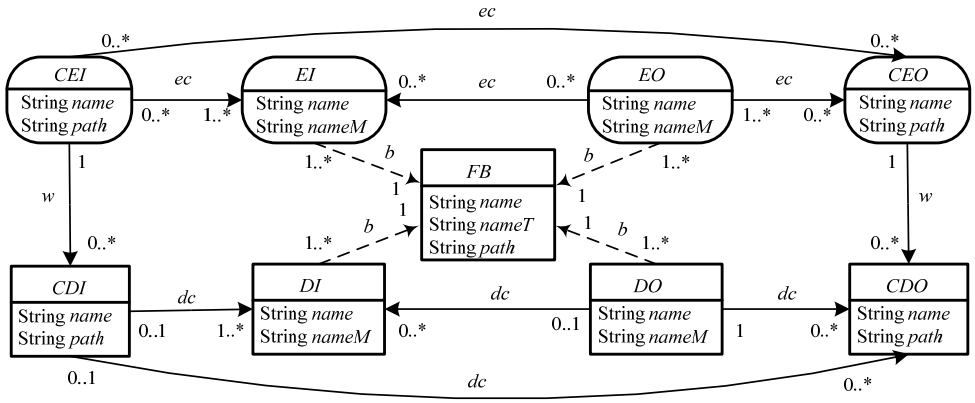


Рис. 9. Метамодель составного ФБ и субприложения

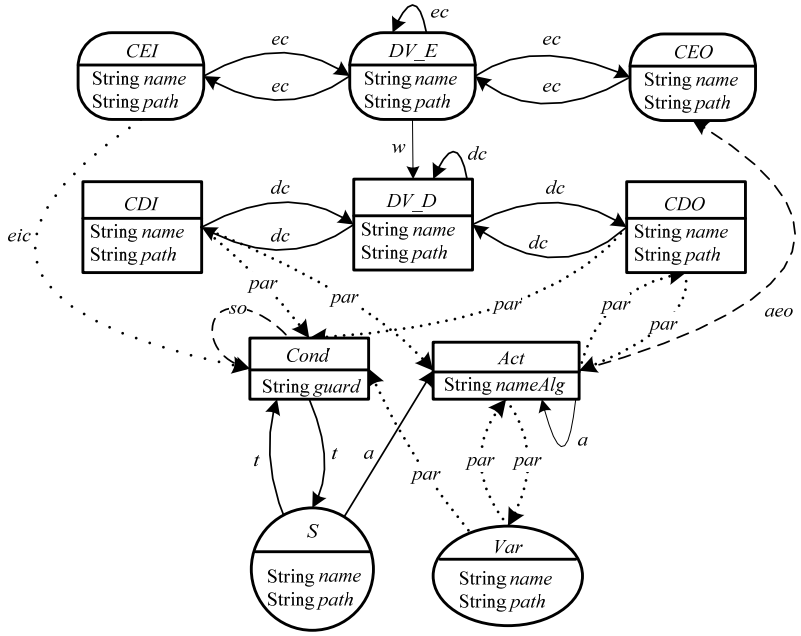


Рис. 10. Метамодель замкнутых систем ФБ уровня 2

7. Правила перехода от многоуровневой структуры систем функциональных блоков к одноуровневой структуре

Переход от многоуровневой структуры системы ФБ к одноуровневой структуре производится в два этапа. На первом этапе (этапе компоновки) рекурсивно осуществляется замена ссылочных экземпляров ФБ на соответству-

ющие им развернутые экземпляры. На втором этапе (этапе встраивания) производится встраивание развернутых экземпляров ФБ в окружающую систему.

Все правила перехода к одноуровневой структуре функционально можно разбить на несколько групп. Классификация правил приведена на рис. 11.

Рис. 11. Классификация правил перехода от многоуровневой структуры систем ФБ к одноуровневой структуре

Библиографический список

1. International Standard IEC 61499. Function blocks for industrial-process measurement and control systems (edition 2.0). Part 1: Architecture / International Electrotechnical Commission. – Geneva, 2005. – 245 p.
2. **Дубинин, В. Н.** Модельно-центрированная методология проектирования распределенных компонентно-базированных информационно-управляющих систем промышленной автоматики / В. Н. Дубинин // Современные информационные технологии : тр. Междунар. науч.-техн. конф. – Пенза, 2013. – Вып. 18. – С. 7–24.
3. **Ehrig, H.** Fundamental theory for typed attributed graph transformation / H. Ehrig, U. Prange, G. Taenzer // Graph Transformation: 2nd Int. Conf. (ICGT 2004). Lecture Notes in Computer Science. – Springer-Verlag, 2004. – Vol. 3256. – P. 161–177.
4. **Sendall, S.** Model transformation: The heart and soul of model-driven software development / S. Sendall, W. Kozaczynski // IEEE Software. Special Issue on Model-Driven Software Development. – 2003. – № 20 (5). – P. 42–45.
5. **Дубинин, В. Н.** Графо-трансформационный подход к синтезу формальных моделей систем функциональных блоков IEC 61499 / В. Н. Дубинин, В. В. Вяткин // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2008. – № 4. – С. 16–26.
6. AGG – A Development Environment for Attributed Graph Transformation Systems. – URL: http://tfs.cs.tu-berlin.de/agg
7. **Dubinin, V.** On Definition of a Formal Model for IEC 61499 Function Blocks / V. Dubinin, V. Vyatkin // EURASIP Journal on Embedded Systems. – 2008. – № 1. – P. 1–10.

References

1. *International Standard IEC 61499. Function blocks for industrial-process measurement and control systems (edition 2.0). Part 1: Architecture*. International Electrotechnical Commission. Geneva, 2005, 245 p.
2. Dubinin V. N. *Sovremennye informatsionnye tekhnologii: tr. Mezhdunar. nauch.-tekhn. konf.* [Modern information technologies: proceedings of the International scientific and technical conference]. Penza, 2013, iss. 18, pp. 7–24.
3. Ehrig H., Prange U., Taenzer G. *Graph Transformation: 2nd Int. Conf. (ICGT 2004). Lecture Notes in Computer Science*. Springer-Verlag, 2004, vol. 3256, pp. 161–177.
4. Sendall S., Kozaczynski W. *IEEE Software. Special Issue on Model-Driven Software Development*. 2003, no. 20 (5), pp. 42–45.
5. Dubinin V. N., Vyatkin V. V. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki* [University proceedings. Volga region. Engineering sciences]. 2008, no. 4, pp. 16–26.
6. *AGG – A Development Environment for Attributed Graph Transformation Systems*. Available at: http://tfs.cs.tu-berlin.de/agg
7. Dubinin V., Vyatkin V. *EURASIP Journal on Embedded Systems*. 2008, no. 1, pp. 1–10.

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

Климкина Людмила Петровна
старший преподаватель, кафедра организации и информатизации производства, Пензенский государственный аграрный университет (Россия, г. Пенза, ул. Ботаническая, 30)

E-mail: ludmila.klimkina@gmail.com

Klimkina Lyudmila Petrovna
Senior lecturer, sub-department of organization and informatization of manufacturing, Penza State Agricultural University (30 Botanicheskaya street, Penza, Russia)

Калачев Андрей Валентинович
аспирант, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: andrei.kalachev@gmail.com

Kalachev Andrey Valentinovich
Postgraduate student, Penza State University (40 Krasnaya street, Penza, Russia)

УДК 681.5

Дубинин, В. Н.

Многоуровневые системы функциональных блоков IEC 61499 и их преобразование к одноуровневым моделям / В. Н. Дубинин, Л. П. Климкина, А. В. Калачев // Известия высших учебных заведений. Поволжский регион. Технические науки. – 2017. – № 1 (41). – С. 16–29. DOI 10.21685/2072-3059–2017-1-2