УДК 519.717

В. Н. Дубинин, В. В. Вяткин

РЕФАКТОРИНГ ДИАГРАММ УПРАВЛЕНИЯ ВЫПОЛНЕНИЕМ СТАНДАРТА *IEC* 61499

В работе рассматривается рефакторинг диаграмм управления выполнением (диаграмм *ECC*) в рамках графо-трансформационного подхода к проектированию распределенных компонентно-базированных систем управления промышленными процессами на основе нового международного стандарта *IEC* 61499. Основной целью рефакторинга является избавление диаграммы *ECC* от условных дуг без событий и потенциально-тупиковых (по условиям) состояний. Приводятся правила преобразования графов для совершения рефакторинга диаграмм *ECC*. Прототип системы рефакторинга реализован в системе трансформации графов *AGG*.

Введение

Международный стандарт *IEC* 61499 определяет архитектуру и языковые средства для построения распределенных компонентно-базированных систем управления промышленными процессами нового поколения [1]. Основой построения систем управления в соответствии с этим стандартом являются функциональные блоки (ФБ). Разработка систем на основе ФБ имеет много общего с разработкой обычного программного обеспечения (ПО) и многие идеи и концепции могут быть заимствованы оттуда.

Одной из передовых современных технологий разработки ПО является технология, основанная на потоке моделей (*Model Driven Engineering – MDE*), использующая модели и их трансформации как первичные артефакты. Трансформация моделей является «душой и сердцем» этой технологии [2]. Перспективным подходом к трансформации моделей является подход, основанный на трансформации графов [3], который бурно развивается и уже нашел применение в *MDE* [4]. Существуют хорошие предпосылки для использования *MDE* и графовых преобразований в проектировании распределенных компонентно-базированных систем управления промышленными процессами на основе нового международного стандарта *IEC* 61499, поскольку модели ФБ могут быть естественно представлены в виде графов [5]. Это относится как к составным ФБ, приложениям и субприложениям, так и к базисным ФБ, основой которых является диаграмма *ECC*.

Важным направлением в области трансформации моделей является рефакторинг моделей. В широком смысле под *рефакторингом* ПО понимается изменение его внутренней структуры без изменения внешнего поведения с целью повышения качества ПО [6]. Рефакторинг определяется на различных уровнях представления ПО – от представления в виде кода и до представления в виде моделей. Рефакторинг является одним из элементов поддержки эволюции программных систем.

В данной работе решается задача рефакторинга диаграмм *ECC* базисных ФБ. Основной целью рефакторинга является избавление диаграммы *ECC* от условных дуг без событий и потенциально-тупиковых (по условию) состояний. Рефакторинг в данном случае во многом основывается на понятии достижимости последовательностей *EC*-акций. Приводятся правила преобразования графов для совершения рефакторинга диаграмм *ECC*. Прототип системы рефакторинга реализован в системе трансформации графов *AGG* [7].

1 Модель диаграммы управления выполнением *ECC*

Диаграмма *ECC* определяет порядок выполнения операций в базисном ФБ и представляет собой диаграмму состояний особого вида [1]. Для решения задач рефакторинга будем использовать упрощенную модель *ECC*, отличную от [8].

Определим диаграмму *ECC* как кортеж:

$$ECC = (S, R, E, C, A, f_E, f_C, f_A, f_P),$$

где $S = \{s_1, s_2,\ldots, s_n\}$ – множество вершин, представляющих *EC*-состояния; $R \subseteq S \times S$ – множество дуг, представляющих *EC*-переходы; $E = \{e_1, e_2,\ldots, e_m\}$ – множество событийных входов; $C = \{c_1, c_2,\ldots, c_k\}$ – множество сторожевых условий, определенных на множествах входных, выходных и внутренних переменных базисного ФБ; $A = \{a_1, a_2,\ldots, a_p\}$ – множество последовательностей *EC*-акций.

Множество дуг R разбивается на классы: R_E – событийных, R_C – условных и R_T – безусловных дуг, $R = R_E \cup R_C \cup R_T$; $R_E \cap R_C \cap R_T = \varnothing$.

Событийная дуга (*E*-дуга) представляет *EC*-переход с событием, условная дуга (*C*-дуга) – *EC*-переход без события, имеющий сторожевое условие, отличное от тождественно истинного, а безусловная дуга (*T*-дуга) аналогично дуге второго типа, но сторожевое условие этой дуги тождественно истинно. В дальнейшем будем обозначать *E*- и *T*-дуги сплошной линией, а *C*-дуги – пунктирной. Над *T*-дугой будем ставить символ «*t*», над *E*-дугой – символ «*e*» (если это необходимо).

f_E: $R_E \to E$ – функция, назначающая *E*-дугам событийные входы.

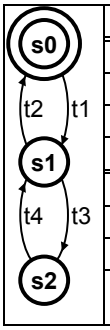
f_C: $R_E \cup R_C \to C$ – функция, назначающая *E*- и *C*-дугам сторожевые условия.

f_A: $S \to A$ – функция, назначающая состояниям последовательности *EC*-акций.

f_P:$R \to D$ – функция, назначающая дугам значения приоритетов, $D = \{d_1, d_2, \ldots\}$ – счетное линейно-упорядоченное множество значений приоритетов. Пусть даны две дуги $r_1, r_2 \in R$. Если $f_P(r_1) = d_i$ и $f_P(r_2) = d_j$ и $i < j$, то считается, что приоритет дуги r_1 выше приоритета дуги r_2. В диаграмме *ECC* стандарта *IEC* 61499 приоритет *EC*-перехода явно не указывается. Для его задания используется позиционный принцип описания *EC*-переходов в *XML*-документе.

2 Модели выполнения *ECC*

В соответствии со стандартом *IEC* 61499 управление выполнением *ECC* осуществляет специальный интерпретатор, диаграмма состояний которого представлена на рис. 1. Как было отмечено в [9, 10] определение интерпретации *ECC* в стандарте является неполным. Оно, например, допускает два различных подхода к оценке *EC*-переходов без событий.



Состояние		Операции
*s*0		
*s*1		Оценить переходы
*s*2		Выполнить акции
Переход	Условие	Операции
*t*1	Запустить *ECC*	Съем входных данных
*t*2	Нет разрешенных переходов	
*t*3	Переход разрешен	
*t*4	Акции завершены	

Рис. 1 Диаграмма состояний интерпретатора *ECC* [1]

В соответствии с первым подходом, *EC*-переход без событий разрешен, если он связан с обработкой какого-то конкретного сигнала, который имел место в ближайшем прошлом, и сторожевое условие этого перехода истинно. Второй подход не связывает *EC*-переход с каким-либо конкретным событием. В этом случае разрешенность *EC*-перехода связана только с истинностью соответствующего сторожевого условия. Назовем сторожевое условие перехода без события в первом случае *пассивным*, а во втором случае – *активным*. В литературе отражены оба подхода. Первый подход представлен в [9]. Второй подход представлен в работе, вводящей модель последовательного выполнения ФБ [10]. В дальнейшем будем рассматривать только первую модель выполнения *ECC*, в которой существует прямая необходимость в рефакторинге диаграмм *ECC*.

Определение 1. Потенциально-тупиковым (по условиям) состоянием (ПТУ-состоянием) в модели выполнения *ECC* с пассивными сторожевыми условиями назовем *EC*-состояние s_i такое, что $\forall j \in \overline{1,n}$ $[(s_i, s_j) \in R \rightarrow (s_i, s_j) \in R_C]$. Иными словами, s_i является ПМУ-состоянием, если все выходящие из него дуги являются условными.

Утверждение. Если интерпретатор *ECC* совершил переход $t2$ (рис. 1) в то время как диаграмма *ECC* находилась в ПТУ-состоянии, то это состояние становится тупиковым. В дальнейшем никакие входные сигналы ФБ не смогут его изменить.

Определение 2. Две *ECC* называются *функционально эквивалентными* (в рамках определенной модели выполнения *ECC*), если при любых последовательностях входных событий и сопутствующих им наборов значений входных переменных обе *ECC* выполняют одни и те же последовательности *EC*-акций.

3 Общий подход к рефакторингу диаграмм *ECC*

Рефакторинг диаграмм *ECC* используется:

1) для избавления от *C*-дуг;

2) для избавления от ПТУ-состояний.

В соответствии с этим будем различать рефакторинги первого и второго типов (рефакторинги 1 и 2). Результаты рефакторинга 2 базируются на результатах рефакторинга 1. Прямую практическую значимость имеет рефакторинг 2. В то же время использование рефакторинга 1 может предоставить разработчику несколько иную точку зрения на разработанную *ECC*, что в ряде случаев (на основе визуального анализа) может помочь ему переосмыслить и перепроектировать диаграмму *ECC*.

Назовем *CT*-сетью диаграммы *ECC* подграф, содержащий дуги только из $R_C \cup R_T$; но не из R_E. В общем случае данный граф будет несвязным. Соответственно, *T*-сетью *ECC* будем называть подграф, содержащий дуги из R_T.

Введем $ES = \{(s, s') \in R_E \mid \exists (s', s'') \in R_C \cup R_T\}$ – множество *E*-дуг, образующих путь длиной 2 с одной из *C*- или *T*-дуг *CT*-сети. Назовем эти дуги источниками. Предполагается, что исходная *CT*-сеть ациклична. Наличие циклов свидетельствует о некорректности *ECC*.

Ниже представлена общая идея избавления *ECC* от событийных дуг. Она основывается на понятии достижимости последовательностей *EC*-акций при интерпретации *ECC*. Пусть (s_0, s_1) – событийная дуга, за которой следует

путь $s_1, s_2, \ldots, s_k$ в *CT*-сети. Каждому *EC*-состоянию s_i ($i = \overline{1,k}$) соответствует последовательность *EC*-акций a_i. В примере на рис. 2 путь составлен только из *C*-дуг, что не меняет сути дела.

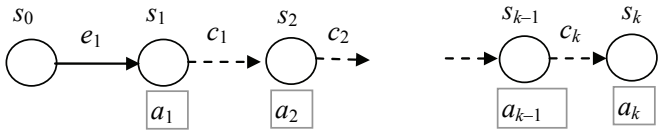


Рис. 2 Путь, состоящий из дуги-источника и *C*-дуг

Этот путь можно заменить одной *E*-дугой (s_0, s_k), сторожевое условие которой составляется из сторожевых условий дуг (c_i, $i = \overline{1,k}$), входящих в путь, и так называемого условия сохранения целевого состояния q (рис. 3). Последовательность *EC*-акций, выполняемых в целевой вершине s_k, определяется как конкатенация (сцепление) последовательностей *EC*-акций вершин, входящих в путь.

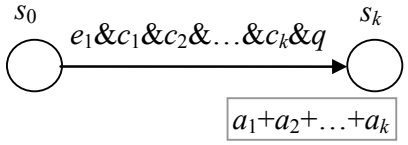


Рис. 3 *E*-дуга, представляющая путь из рис. 2

Условие сохранения некоторого состояния s_k определяется как конъюнкция отрицаний сторожевых условий исходящих *C*-дуг: $\prod_{(s_k,s_j)\in R_C} \overline{f_C(s_k,s_j)}$, здесь операция конъюнкция обозначена символом П. Например, для состояния s_k на рис. 4 условие сохранения состояния равно $\overline{c_{k+1}} \,\&\, \overline{c_{k+2}} \,\&\, \ldots \,\&\, \overline{c_{k+n}}$.

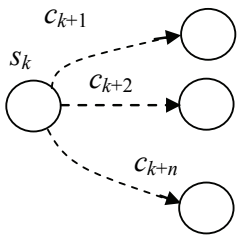


Рис. 4 Пример для иллюстрации вычисления условия сохранения состояния

Если из состояния s_k исходит *T*-дуга, то это состояние является транзитным и переход в него невозможен, поскольку интерпретатор *ECC* все время будет «проскакивать» через это состояние.

Если в целевое состояние s_k пути входят какие-то другие дуги, то путь $s_1, s_2, \ldots, s_k$, начинающийся из дуги (s_0, s_1), следует заменять двумя дугами (s_0,

s_{k-1}) и (s_{k-1}, s_k), первая из которых в целом идентична *E*-дуге из рис. 3, а вторая дуга является *T*-дугой. Необходимость второй дуги определяется тем, что входящие в целевое состояние s_k дуги определяют альтернативные пути, которым соответствует суммарная последовательность *EC*-акций, в общем случае отличная от последовательности *EC*-акций, определяемых рассматриваемым путем. Поэтому приписывание целевому состоянию s_k последовательности *EC*-акций какого-то одного из путей было бы неверным. Выход из этого положения – приписать *EC*-акции пути (за исключением *EC*-акций целевой вершины) промежуточной вершине s_{k+1} и соединить ее безусловной дугой с целевой вершиной (рис. 5). Назовем вершину s_{k+1} представителем вершины s_k, поскольку большинство *EC*-акций, которые выполнялись бы в состоянии s_k, выполняются именно в этой промежуточной вершине.

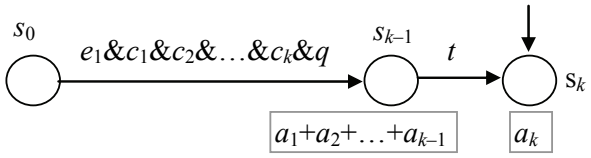


Рис. 5 *E*- и *T*-дуги, представляющие путь из рис. 2

Назовем *связыванием дуги* $r_i \in ES$ *с вершиной* s_k *CT*-сети операцию нахождения всех путей в *CT*-сети, ведущих из конца дуги r_i в вершину s_k и генерации на их основе *E*-дуг или пар дуг типа (*E*, *T*) в соответствии с изложенным выше методом. В общем случае результатом такой операции является структура, представляющая гамак. Каждой *E*-дуге при этом назначен один и тот же событийный вход. Следует отметить, что связывание дуги $r_i \in ES$ с вершиной s_k *CT*-сети не всегда возможно.

Назовем *связыванием дуги* $r_i \in ES$ с *CT-сетью* операцию связывания этой дуги со всеми вершинами *CT*-сети. Для полного избавления *ECC* от *C*-дуг необходимо связать все дуги из множества *ES* с соответствующей *CT*-сетью и далее все *C*-дуги удалить. Можно утверждать, что любую ациклическую *CT*-сеть можно преобразовать к виду, свободному от *C*-дуг. Полученную в результате таких преобразований *T*-сеть в совокупности с соответствующими *E*-дугами можно назвать графом достижимости наборов *EC*-акций в исходной *CT*-сети. Очевидно, что преобразованная в соответствии с предложенным методом *ECC* будет эквивалентна исходной.

При проведении рефакторинга важно не только получить новую структуру *ECC*, вычислить сторожевые условия и наборы выполняемых *EC*-акций, но и определить приоритеты дуг в обновленной *ECC*. Предлагается использовать составные приоритеты (в виде кортежа) с заданным на них лексикографическим порядком $\prec$. Составной приоритет формируется как конкатенация приоритетов дуг пути от начальной вершины до целевой. Чем ближе к началу пути находится дуга, тем большее влияние оказывает она на «суммарный» приоритет. Преимуществом составных приоритетов является легкость их вычислений. На рис. 6 приведен пример *ECC* в виде бинарного дерева с одной *E*-дугой в качестве «движущей силы». В исходной *ECC* используются нормализованные приоритеты. На рис. 7 представлена преобразованная *ECC*. Вычисленные составные приоритеты записаны под соответствующими дугами.

Как видно из рис. 6, дуга (s_0, s_1) – самая приоритетная, поскольку $\forall pr \in \{(1, 2), (1, 3), (2, 1), (2, 2)\}[(1, 1) \prec pr]$.

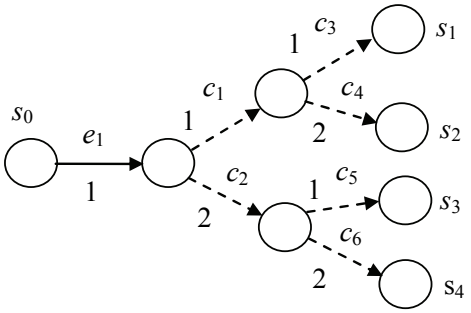


Рис. 6 Диаграмма *ECC* в виде бинарного дерева

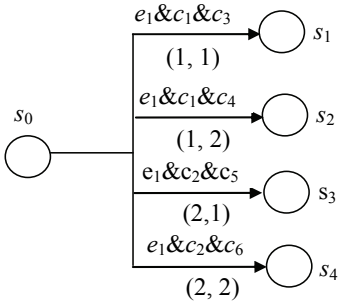


Рис. 7 Преобразованная диаграмма *ECC* (из рис. 6) с составными приоритетами дуг

Существует два типа рефакторинга в зависимости от того, из каких состояний производится «трассировка сигнала». В первом случае активной дугой является *E*-дуга, выходящая из любого состояния, во втором случае – только та *E*-дуга, которая выходит из состояния, достижимого из начального. Второй вариант назовем рефакторингом с учетом достижимости из начального состояния.

4 Рефакторинг на основе графотрансформационного подхода

Ниже предлагается подход к решению задачи рефакторинга на основе локальных эквивалентных преобразований *ECC* с использованием системы перезаписи типизированных атрибутных графов. Одно эквивалентное преобразование может выражаться в общем случае применением не одного, а нескольких правил. Процесс локальных эквивалентных преобразований графовой модели *ECC* проходит в рамках вычислений, описанных в разделе 3.

Основные правила трансформации связаны с обработкой пары смежных дуг (s_i, s_j) и (s_j, s_k) и формированием на их основе новой прямой дуги, ведущей в состояние s_k или в его представителя. Смыслом большинства правил является построение множества достижимых (некоторым сигналом) наборов *EC*-акций путями длиной 2. Дуги, входящие и выходящие из вершин s_i, s_j и s_k, представляют контекст применения правила.

Все правила можно разделить на следующие группы:

1) правила предварительной обработки (корректировки);

2) правила наращивания графа;

3) правила очистки графа.

Правила первой группы приводят исходную *ECC* к некоторой нормализованной форме. Примерами правил данного типа являются правило слияния параллельных *C*-дуг, правило удаления мертвых *E*-дуг, правило удаления мертвой неприоритетной *T*-дуги

Правила наращивания графа по выполняемой функции делятся на несколько групп. Внутри группы правила различаются только контекстом. На рис. 8–11 схематично представлены основные группы правил данного типа в виде некоторых шаблонов. Контекстные состояния представлены маленькими кружками. Тип контекстных дуг не уточняется. Крест на контекстной дуге в левой части правила означает, что дуга запрещена. Таким путем кратко представляются *NAC*-условия. Черточка на дуге в правой части правила озна-

чает, что дуга является использованной и в последующем будет удалена. Для *E*-дуги вычисляется два условия: условие достижения целевого состояния (включающее также имя событийного входа) и условие распространения сигнала за целевое состояние. Первое условие пишется над дугой, а второе – под дугой. Полное текущее условие для *E*-дуги определяется как конъюнкция первого условия и отрицания второго условия.

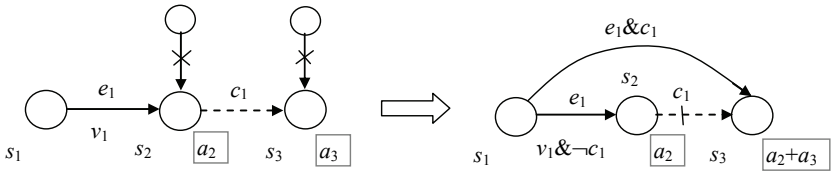


Рис. 8 Правило распространения сигнала (на линейном участке)(*R*1)

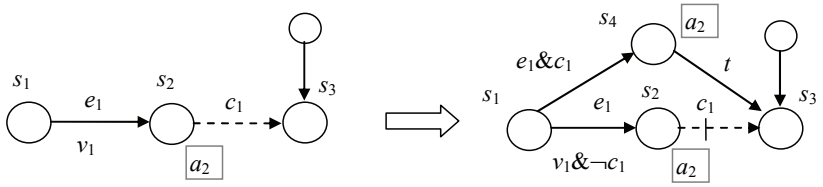


Рис. 9 Правило вхождения в соединитель (*R*2)

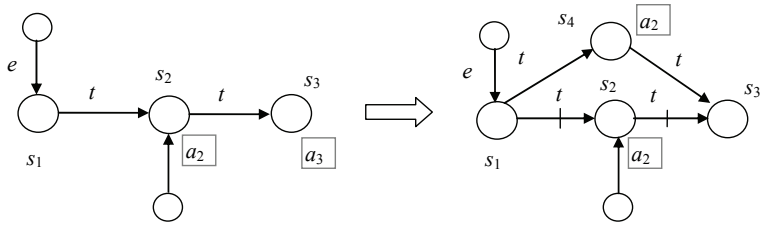


Рис. 10 Правило обхода соединителя (*R*3)

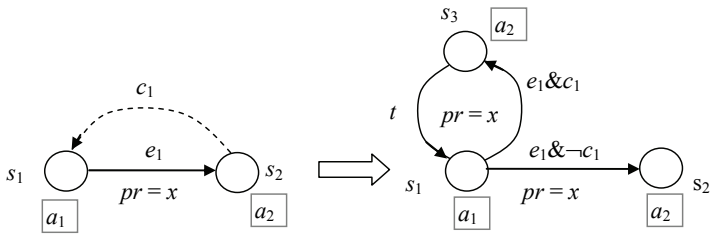


Рис. 11 Правило избавления от обратной *C*-дуги (*R*4)

Как правило, вновь создаваемая (дочерняя) дуга имеет двух родителей – пары следующих друг за другом смежных дуг. При этом приоритет дочерней дуги полагается равным конкатенации приоритетов родительских дуг, причем первым идет приоритет дуги, стоящей в этой паре первой.

При расщеплении узла (например, при обходе соединителя) *E*-дугам не выделяется отдельная вершина, они используют имеющуюся вершину-соединитель.

В правиле избавления от обратной *C*-дуги (рис. 11) в правой части правила, в отличие от большинства правил, имеются две дочерние дуги (s_1, s_3) и (s_1, s_2), они имеют одинаковый приоритет, равный приоритету *x* родительской дуги. Это не имеет значения, т.к. они являются взаимоисключающими по условию c_1.

В отличие от правил первой и второй групп, правила очистки графа являются специфическими для каждого из типов проводимого рефакторинга. В качестве примера правил данной группы можно назвать правило удаления *E*-дуги и правило удаления изолированной вершины.

Для иллюстрации процесса рефакторинга рассмотрим простую диаграмму *ECC* (рис. 12) и ее преобразование. Упрощенно процесс рефакторинга 1 определяется цепочкой правил *R*1, *R*1. Упрощение (для облегчения понимания сути) заключается в том, что использованные *C*-дуги уничтожаются непосредственно в правиле *R*1. После первого применения правила *R*1 получается промежуточная *ECC*, представленная на рис. 13. Как можно заметить, она эквивалента исходной *ECC*. После второго применения правила *R*1 получается результирующая для рефакторинга 1 *ECC* (рис. 14). Она также эквивалентна исходной *ECC*. В результирующей *ECC* ПТУ-состояния представлены в виде тупиковых вершин s_1 и s_2.

Процесс рефакторинга 2 описывается цепочкой правил *R*1, *R*1, *R*5, *R*6. Результирующая *ECC* для этого случая представлена на рис. 15. Эта *ECC* не эквивалентна исходной *ECC*, поскольку в ней отсутствуют ПТУ-состояния s_1 и s_2. Это приводит к тому, что, например, при входном сигнале e_1 истинности условия c_1 и ложности условия c_2 (кратко $ei_1\&c_1\&\neg c_2$) в результирующей *ECC* не выполняется ни одна *EC*-акция, в то время как в исходной *ECC* выполняются последовательности *EC*-акций a_2 и a_3. Несмотря на то, что исходная и результирующая диаграммы *ECC* не эквивалентны, результирующая диаграмма *ECC* на рис. 15, скорее всего, соответствует замыслам разработчика, учитывая то, что он хорошо осведомлен об особенностях модели выполнения и не допускает возможности того, что *ECC* останется в «замороженном» состоянии. Следует отметить, что в примерах на рис. 12–15 приоритеты дуг не имеют никакого значения вследствие взаимоисключающих сторожевых условий.

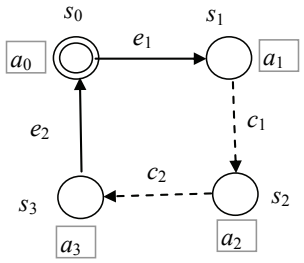


Рис. 12 Диаграмма *ECC*

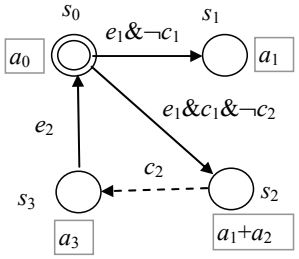


Рис. 13 Промежуточная диаграмма *ECC*

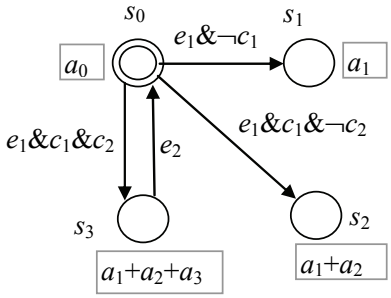


Рис. 14 Результат рефакторинга 1 диаграммы *ECC*, представленной на рис. 12

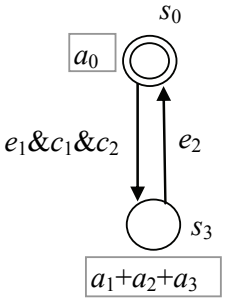


Рис. 15 Результат рефакторинга 2 диаграммы *ECC*, представленной на рис. 12

Более сложный пример диаграммы *ECC* и результат ее рефакторинга второго типа представлен на рис. 16. Нормализованные приоритеты дуг в виде чисел поставлены рядом с дугами. Для краткости для результирующей *ECC* опущены приоритеты тех дуг, которые являются единственными выходящими из вершины.

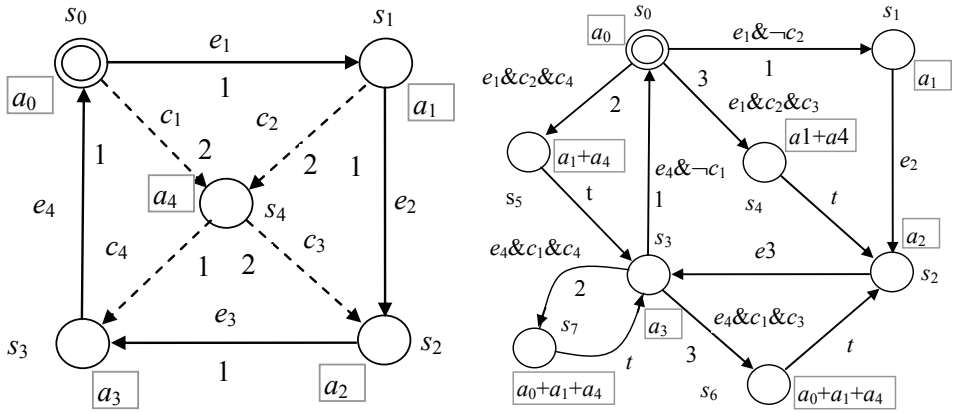


Рис. 16 Диаграмма *ECC* (слева) и результат ее рефакторинга 2 (справа)

Прототип системы рефакторинга *ECC* был разработан с помощью системы трансформации графов *AGG*. Система *AGG* представляет основанную на правилах среду визуального программирования, использующую алгебраический подход к трансформации графов [7]. Правила перезаписи графов могут содержать *NAC*- и контекстные условия. Разработана метамодель диаграммы *ECC*, используемая в прототипе системы рефакторинга, в виде типизированного графа, сформированного в *AGG*. Система рефакторинга содержит около 35 правил.

Подробные сведения из теории трансформации графов можно найти в [3].

Заключение

В данной работе представлен графо-трансформационный подход к рефакторингу диаграмм *ECC*. В рамках данного подхода разработана графовая модель *ECC*, проанализированы модели выполнения *ECC* и сформулированы цели рефакторинга, представлен общий подход для проведения рефакторинга в терминах графов и, наконец, приведены основные правила перезаписи графов системы рефакторинга. Прототип системы рефакторинга реализован в системе трансформации графов *AGG*.

Список литературы

1. Function blocks for industrial-process measurement and control systems. – Part 1: Architecture, International Electrotechnical Commission. – Geneva, 2005.
2. **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.
3. **Ehrig, H.** Handbook of Graph Grammars and Computing by Graph Transformations / H. Ehrig, G. Engels, H.-J. Kreowski, G. Rozenberg. – World Scientific, 1999. – Vol. 1.
4. **Grunske, L.** Graph Transformation for Practical Model Driven Software Engineering / L. Grunske, L. Geiger, A. Zundorf, N. V. Eetvelde, P. V. Gorp, D. Varro // Model-driven Software Development. – 2005. – Springer. – P. 91–118.

5. **Дубинин, В. Н.** Построение поисково-трансформационных систем для поддержки проектирования компонентно-базированных систем промышленной автоматики / В. Н. Дубинин, В. В. Вяткин // AIS'05 и CAD-2005 : труды Международных научно-технических конференций. – Дивноморское ; М. : Физматлит, 2005. – Т. 2. – С. 30–35.
6. **Mens, T.** On the Use of Graph Transformations for Model Refactoring / T.Mens // Lecture Notes in Computer Science. Generative and Transformational Techniques in Software Engineering. – 2006. – Vol. 4143. – P. 219–257.
7. **Taenzer, G.** AGG: A Tool Environment for Algebraic Graph Transformation / G. Taenzer // Lecture Notes in Computer Science. – 2000. – Springer. – Vol. 1779. – P. 481–490.
8. **Dubinin, V.** Towards a Formal Semantics of IEC 61499 Function Blocks / V. Dubinin, V. Vyatkin // 4th IEEE International Conference on Industrial Informatics (INDIN'2006). – Singapore, 2006. – P. 6–11.
9. **Sünder, C.** Usability and Interoperability of IEC 61499 based distributed automation systems / C. Sünder, A. Zoitl, J. H. Christensen, V. Vyatkin, R. Brennan, A. Valentini, L. Ferrarini, K. Thramboulidis, T. Strasser, J. L. Martinez-Lastra, F. Auinger: // 4th IEEE Conference on Industrial Informatics (INDIN 2006). – Singapore, 2006. – P. 31–37
10. **Vyatkin, V.** Sequential Axiomatic Model for Execution of Basic Function Blocks in IEC61499 / V. Vyatkin, V. Dubinin // 5th IEEE Int Conf. on Industrial Informatics (INDIN'2007). – Vienna, 2007. – P. 1183–1188.