Управление большими системами
Специальный выпуск 30.1 «Сетевые модели в управлении»

УДК 021.8 + 025.1
ББК 78.34

СЕТЕЦЕНТРИЧЕСКОЕ УПРАВЛЕНИЕ НА СЕТЯХ ПЕТРИ В СТРУКТУРИРОВАННОЙ ДИСКРЕТНО-СОБЫТИЙНОЙ СИСТЕМЕ

Амбарцумян А. А.[1]
(*Учреждение Российской академии наук*
Институт проблем управления РАН, *Москва*)

На основе концептов сетецентрического управления разработана методика синтеза управляющей сети Петри для систем автоматизации, работающих в реальном времени. Методика использует для анализа функциональности и согласованности модель СДСС – структурированная дискретно-событийная система. Определен состав модели, предложена техника анализа управляемости и моделирования объекта сетью Петри процесса. Разработана техника анализа сети Петри процесса и метод синтеза супервизора – управляющей сети Петри, обеспечивающей совместно с сетью процесса выполнение спецификации СДСС [2].

Ключевые слова: системы управления, сетецентрические системы, логическое управление, дискретно-событийные системы, событийные модели, сети Петри.

1. Введение

Определяющей тенденцией современного развития систем информатизации сложных технологических комплексов являет-

---

[1] *Александр Артемович Амбарцумян*, *доктор технических наук*, *профессор* (*ambar@ipu.ru*).
[2] *Работа выполнена при частичной поддержке РФФИ* (*проект 09-08-00559-а*).

ся интегрирование различных задач жизнеобеспечения технологических процессов с их системами автоматизации, работающими в реальном времени (САРВ), в комплексную систему путем охвата автономных компонент сетевой структурой. Функциональность современных САРВ интегрирует задачи контроля и управления: технологическими процессами, безопасностью производства, защитой оборудования, технологической транспортной системой, обслуживанием и ремонтами, экологическим состоянием среды и т. д. [2, 15]. Проблемы управления, обусловленные запросами обеспечения эффективного совместного взаимодействия разнородных систем, характерны для многих сложных технологических комплексов энергетики, добывающих отраслей и транспортных систем. Проектирование таких САРВ достаточно трудоемко и требует значительных временных и человеческих ресурсов. Легко просматривается аналогия интегрированных и сетецентрических систем.

Характерная особенность сетецентрических («*network-centric*») систем выражается в способности каждой территориально-распределенной компоненты (узла) действовать в направлении достижения общей цели и иметь равные возможности к доступу информации, необходимой для осуществления функций и достижения цели, вне зависимости от расположения этой информации в системе [7]. Создание сетецентрических систем стало реальностью с развитием современных *IT*-инструментов. Однако наиболее известные успехи сетецентрических технологий в управлении военными действиями армии США стали возможны благодаря подготовленной уже ранее доктрине сетецентрического стратегического управления (ССУ) военными действиями, известной как «*The Strategic Theory of John Boyd* (*Boyd`s OODA loop*)» [9]. Важнейшая черта ССУ: *подготовка в сети данных для потребителей в соответствии с их задачами* по принципам цикла *OODA* (сбор данных под задачи, оценка ситуации, принятие решения и его реализация).

Вместе с тем в проектировании интегрированных САРВ системные решения всегда являются творческим озарением проектантов, а не результатом каких либо аналитических построений, и, как правило, эти решения основаны на интуитивных методах и искусстве системотехника [2, 15, 16]. Вследствие этого структура сетевых решений в основном формируется на основе естественной топологии расположения автономных компонент и выполняет функциональность, обусловленную только потребностями транспортирования данных. Поэтому, по мнению автора, концепты (методы) управления, ориентированные на вовлечение в процесс достижения цели (выполнения миссии) всех компонент системы, и доступность любой информации, требуемой для принятия управленческого решения, для систем промышленной автоматизации нуждаются в развитии.

Подходом к проектированию управления оборудованием с дискретным поведением, наиболее близко соответствующим концепции ССУ, на наш взгляд, является дискретно-событийное моделирование, включая парадигму супервизорного управления [18, 20].

Отличительной особенностью теории супервизорного управления в рамках дискретно-событийной системы (ДСС) является моделирование управляемого объекта тройкой компонент:

- G – собственно объект управления, рассматриваемый как генератора языка $L(G)$;
- спецификация K – это ограничения и востребованная функциональность G, задаваемая тоже как язык $K \subseteq L(G)$;
- супервизор S – управляющая компонента в ДСС обеспечивающая поведение G, в соответствии с ограничениями (спецификацией) K.

При этом S должен быть неблокирующим – не содержать тупиков и «живых циклов» (ловушек).

В рамках определенного триплета $<G, K, S>$ в теории супервизорного управления поставлена и решена задача формального синтеза неблокирующего S по $L(G)$ и K. Постановка и решение

задачи синтеза создает теоретический базис для гарантированного проектирования событийного (часто – автоматного) управления, принципиально отличающийся от традиционного подхода к проектированию логического управления как расшифровки двух черных ящиков – объекта управления и алгоритма управления, интуитивно соответствующих друг другу. Справедливости ради нужно заметить, что очень близкая постановка была известна в русскоязычной литературе по теории конечных автоматов еще в 70-е годы (см., например, работы ученых школы члена-корреспондента М.А. Гаврилова [4, 5] и наиболее близко – исследования в коллективе профессора А.А. Таля [8, 11, 13]). В работе [11] для пары «черный ящик – объект управления» и «черный ящик – система управления» введена конструкция «взаимодействующие автоматы»; определены понятия управляемости и наблюдаемости, сформулированы и исследованы условия их выполнения и синтез управляющего автомата по заданному поведению автомата – объекта. Обобщение этих и многих других результатов приведено в монографии [8]. Однако концентрация исследования на уже вполне определенном поведении управляемого объекта методологически значительно сложнее, чем определение поведения пары: ничем не ограниченного объекта и требований (ограничений) к его поведению. Возможно, по этой причине практического приложения за этими исследованиями не последовало.

Поток публикаций по ДСС в зарубежной литературе огромен. Укажем пионерскую работу [24] и обобщающую монографию [18]. Две основные трудности в практическом использовании ДСС-моделирования: первая заключается в размерности модели объекта управления – G (для реальных задач число состояний конечного автомата порядка несколько тысяч); вторая – в мультипликативной зависимости сложности супервизора, от сложности объекта G и сложности спецификации K. Для преодоления этих затруднений разработаны методы модульного синтеза супервизора по G. При этом предложены различные схемы управления: дизъюнктивная, конъюнктивная, иерархиче-

ская, обобщенная (см. первую известную работу [26], развитие и обобщение в работе [28]). Разработаны методы синтеза по модульному описанию $G = <G_1, G_2, \ldots, G_n>$ и модульному заданию спецификации $K = <K_1, K_2, \ldots, K_n>$ модульного супервизора S [19, 14, 21]. Однако сложность модульного синтеза и слабое соответствие структуры исходных спецификаций результату делают предложенные методы малопривлекательными для практического применения. Сохранение исходной модульности для отдельных классов ДСС достаточно эффективно обеспечивается использованием сетей Петри (СП) как аппарата описания и синтеза супервизорного управления в ДСС (работы [20, 23, 27] и многие другие).

Использование сетей Петри как аппарата анализ и синтеза управления для систем промышленной автоматизации известно и раннее. В работах 70-х годов – М. Хэка и многих других авторов (обзор можно найти в [25]) – исследовались методы анализа поведения сложных производственных систем на сетях Петри (СП). В русскоязычной литературе известны исследования С.А. Юдицкого [12] по иерархическим СП для задачи описания и анализа производственных систем.

Характерна следующая постановка исследований 70-80 гг. прошлого века по СП для управления промышленными системами: задана СП S, предполагается, что она определяет алгоритм управления. Задача синтеза – отвечает ли S определенным требованиям (безопасность, живучесть и т. п.) и если да, то, как преобразовать алгоритм, заданный S, в иной вид (конечный автомат, программу или устройство).

В ДСС постановка задачи управления кардинально отличается: дана СП S_1, моделирующая ничем не ограниченное поведение объекта управления G. Определены ограничения на поведение G – спецификация K. Задача: существует ли СП S_2, дополняющая S_1 (говорят: встроенная в контур обратной связи по определенным правилам) такая, что совместно (S_1 и S_2) удовлетворяют K. Если S_{2c} существует, то как эту сеть синтезировать.

Использование СП как аппарата моделирования ДСС и синтеза супервизора (работы [20, 23] и многие другие) ориентированы, прежде всего, на модульный синтез для отдельных классов ДСС. Метод, предложенный в работе [23], основан на использовании анализа запрещенных состояний и, в общем случае, синтез сети супервизора (как дополнение сети процесса) решается методами целочисленного программирования. Методы в работах [20, 27] основываются на матричном представлении СП и используют технику инвариантов позиций. Эти методы дают хорошие решения для управления ресурсным взаимодействием компонент ДСС (выполнение ограничений на количество одновременно активных компонент или задействованных ресурсов), но эта техника неэффективна при задании спецификации в виде порядка срабатывания всех переходов. Однако для САРВ характерно задание спецификации именно в виде требуемой последовательности операций актуаторов.

Особенности интеграции на принципах сетецентризма приложений САРВ требуют сохранить уже имеющуюся систему (устройство) управления автономных компонент (АК) или (если его нет) сконструировать автономно управление операциями АК. А вот их взаимодействие (последовательность выполнения операций) необходимо реализовать отдельным модулем, который выполняет функции сети связи и реализует логику координации отдельных компонент в соответствии с текущей целью совместного функционирования (*подготовка в сети данных для потребителей в соответствии с их задачами*).

Имеется несколько резонов в такой постановке:

Во-первых, легко реализуется развитие системы. К изменениям отдельных компонент предъявляется лишь требование сохранения операций (команд) взаимодействия. Добавление новых компонент затрагивает только модуль координации – супервизор.

Во-вторых, повышается живучесть системы.

В-третьих, смена стратегии (общей цели) затрагивает только супервизор, что может быть реализовано сменой «матрицы» взаимодействия.

В работах [1, 3] предложена модель структурированной дискретно-событийной системы СДСС, в рамках которой моделирование проводится на двух уровнях. Первый – это компоненты СДСС – актуаторы $G = (G^1, G^2, \ldots, G^n)$. Поведение каждой компоненты G^i определяется конечным автоматом – генератором языка $L(G^i)$. Второй уровень – это спецификация требуемого поведения СДСС в языке команд и условий K (являющегося проекцией $L(G)$). Получены условия управляемости и предложен метод синтеза супервизора, которые выполняются с линейной сложностью от сложности спецификации. При этом, несмотря на модульность задания СДСС, супервизор конструируется как единый модуль второго уровня управления.

В настоящей работе (в развитие работ [1, 3]) ставится следующая задача: в рамках ДСС разработать методику анализа и проектирования супервизора, ориентированную на сохранение структурного подобия результатов проектирования и исходных описаний поведения объекта управления. Сохранение структурного подобия предлагается обеспечить использованием для управления операциями АК их моделей, а на уровень супервизора перенести только передачу управления между модулями, представляющими АК. В методике предлагается данные о ДСС последовательно формировать в моделях: набор конечных автоматов (этап описания данных о структуре объекта), строки команд (событий) на исполнение операций оборудованием (этап формирование требований технологов на последовательности операций); структура выполнении команд АК и последовательностей операций АК – сети Петри (этап системного аналитика и реализации системы).

Структура работы. Во втором разделе мы приводим базовые понятия и результаты, которые используются в предлагаемой методике. В третьем разделе излагается собственно методика проектирования супервизорного управления. В

заключении кратко характеризуется вклад данной работы в проблему проектирования супервизорного управления для систем автоматизации.

2. Базовые понятия. Результаты, на которые опирается данная работа

Настоящая работа основывается на модели «структурированная дискретно-событийная система» (СДСС), предложенной в [1]. Разработка СДСС мотивирована двумя особенностями оборудования с точки зрения ДСС-моделирования.

Во-первых, для дискретного оборудования характерно следующее разбиение множества событий E. Традиционное для теории ДСС разбиение на E_{uc} и E_c – множества управляемых и неуправляемых событий, и E_w – множество ожидаемых событий. События из E_w моделируют состояния (положения) исполнительных механизмов (актуаторов) или компонент объекта. Эти события нельзя блокировать супервизором как управляемые события из E_c. Однако их появление ожидаемо, как отклик на события из E_c, подтверждающий факт выполнения команд на актуаторы.

Во-вторых, поведение каждого актуатора G^i моделируется языком $L(G^i)$ из слов над $E^i = \{E_w{}^i \cup E_c{}^i\}$, а спецификация требуемого поведения формулируется как язык K из событий $E_d = \{E_c \cup E_{uc}\}$ – множества событий, моделирующего совокупности команд и условий их применения. Учет этих особенностей позволяет получить ряд преимуществ как в определении ДСС, так и формулировании условий управляемости и синтезе супервизора.

В рамках СДСС моделирование проводиться на двух уровнях. Первый – это компоненты ДСС – актуаторы $G = (G^1, G^2, \ldots, G^n)$ с множествами E^i событий, каждое из которых структурируется на $E^i = \{E_w{}^i \cup E_c{}^i\}$ и множеством E_{uc} – общих неуправляемых событий. Поведение каждой компоненты G^i определяется конечным автоматом-генератором языка $L(G^i)$,

задаваемым $G^i = \left\langle Q^i, E^i, \delta_i, \Gamma^i, Q_m^i, q_0^i \right\rangle$ – множествами состояний, событий, функциями переходов и допустимых событий, множеством обязательно достижимых состояний и начальным состоянием соответственно. Второй уровень – это спецификация требуемого поведения СДСС как язык $K \subseteq E_d^*$ (где $E_d{}^*$ – множество всевозможных слов любой, но конечной длины из событий $E_d = E_c \cup E_{uc}$). Язык K задается автоматом $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m{}^h, q_0)$ как множество строк, определяющих требуемые спецификации. Язык K назван языком директивных спецификаций (*язык* лент технологических спецификаций). ДСС с перечисленным структурированием названа *ДСС с углубленным разделением событий.*

Система, состав неуправляемой части которой задан ДСС с углубленным разделением событий, требуемое поведение определено языком спецификаций $K \subseteq E_d^*$ $(K \neq \emptyset)$, и обеспеченная супервизором S так, что выполняется K, в работе [1] названа ***структурированной дискретно событийной системой* (*СДСС*)**. Выполнение K предполагает, что проекция $P_{Ed}(L(S / G)) = K$, т. е. K будет совпадать с проекцией по E_d языка $L(S/G)$, генерируемого S/G – объектом G под контролем супервизора S.

В работе [1] сформулированы свойства СДСС и разработана процедура, названная алгоритм эксперимента $\Re(s)$, которая является основным приемом исследования совместного поведения G и H. Результат эксперимента над G строками $s \in K$, такими, что $\delta_k(q_0, s)!$[3], имеет два исхода $\Re(s) \in \{True, False\}$. Если все эксперименты успешны, то G и H согласованы и супервизор S, с которым выполняется K, существует.

При этом предполагается, что G обладает собственным контроллером, обеспечивающим генерацию управляющих

[3] ! означает, что строка s допустима для начального состояния q_0.

событий, а супервизор лишь обеспечивает блокирование тех управляющих событий, появление которых противоречит спецификации. Для моделирования ДСС с пассивными актуаторами (а САРВ именно таковыми и являются) в теории ДСС используется модель с форсируемыми управляемыми событиями [27], в которой управляемые события генерирует (форсирует) супервизор. Существование супервизора для структурированных ДСС с форсируемыми событиями исследованы в работе [3]. В этой работе введено понятие корректного ветвления в графе переходов автомата *H*, задающего язык спецификации *K*. Содержательно: ветвление корректно, если все события, взвешивающие в ветвлении ребра графа *H*, либо являются неуправляемыми, либо выбор предопределяется предшествующими событиями (в каждое ребро в ветвлении ведут из начального состояния автомата *H* отличающиеся пути).

Условия управляемости СДССф – СДСС с форсируемыми управляемыми событиями – определены в *теореме управляемости*.

Теорема управляемости СДССф. *Пусть дан* $G = \left\langle G^1, G^2, ..., G^n \right\rangle$ *с углубленным разделением событий, у которого* $E = \{E_w \cup E_c \cup E_{uc}\}$, *все* $\boldsymbol{E_c}$ *форсируемы,* $E_d = \{E_c \cup E_{uc}\}$, $K \subseteq E_d^*$ $(K \neq \emptyset)$ *и K определяется автоматом H. Неблокирующий супервизор S, такой что* $P_{E_d}(L(S/G)) = K$ *существует, тогда и только тогда, когда для любой* $s \in K$ (*такой, что* $\delta_h(q_0, s)!$) $\Re(s) = True$ *относительно* $G = \left\langle G^1, G^2, ..., G^n \right\rangle$ *и все ветвления в графе переходов H корректны.*

Первая часть предлагаемой методики – анализ существования неблокирующего супервизора *S*, основывается на проверке сформулированных в теореме условий существования *S* и определении $Q = \{Q_1, Q_2, ..., Q_r\}$ – множество достижимых векторов-состояний компонент $G = \left\langle G^1, G^2, ..., G^n \right\rangle$.

Синтез супервизора основан на анализе допустимых строк в языках $L(G^i)$ и $K \subseteq E_d^*$, заданных автоматами $G = \langle G^1, G^2, ..., G^n \rangle$ и $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m{}^h, q_0)$. Специфика промышленных объектов такова, что логика отдельных компонент объекта довольно прозрачна. Графы переходов, задающие $L(G^i)$ и $K \subseteq E_d^*$, – слабответвящиеся и, как правило, задают ограниченное количество циклически выполняемых последовательностей. Поэтому сами строки и их последовательности достаточно просто извлекаются из анализа последовательности ребер графа переходов. Такие последовательности довольно просто описываются на языке ЛСА – логических схем алгоритмов хорошо изученном в работах Лазарева В.Г. и Баранова С.И. [10, 17]. В данной работе мы используем упрощенный вариант ЛСА для описания последовательности строк событий. В СДСС под логической структурой последовательности строк (ЛСПС) будем понимать представление графа переходов конечного автомата в виде строк символов состояний и событий этого графа. Правила перехода от графа переходов к ЛСПС и обратно довольно просты и аналогичны описанным в [10, 17] правилам перехода от граф-схем алгоритмов к ЛСА. Поэтому ограничимся их изложением на концептуальном уровне. Каждая строка ЛСПС соответствует простому пути графа переходов и составляется из имен состояний и событий, взвешивающих ребра, в порядке их следования в этом пути. Следование строк соответствует порядку обхода простых путей в графе переходов и представляется стрелочками с индексами. Стрелочка вверх с индексом $\uparrow^n$ – исходящее ребро, заканчивающее простой путь (после события, взвешивающего это ребро); стрелочка вниз с индексом $\downarrow^n$ – входящее ребро (перед состоянием, в которое входит помеченное раннее стрелкой исходящее ребро). Индексы стрелок однозначно связывают начало и конец перехода между строками. Если после некоторого состояния имеет место ветвление (исходящих ребер более одного), то ветвление заканчивает строку, а события, соответствующие этим ребрам, с верхними стрелочками заключается в

фигурные скобки. На рис. 1 представлен граф переходов механизма G^3, стрелочками помечены ребра графа, разрыв которых выделяет простые пути для построения строк ЛСПС. Для графа переходов G^3 ЛСПС: $\downarrow^1 q_1\ e_{3\text{-}1}\ q_2\ e_{3\text{-}2}\ q_3\ e_{3\text{-}3}\ \downarrow^4 q_4\{e_{3\text{-}8}\ \uparrow^2,\ e_{3\text{-}4}\ \uparrow^3\}$; $\downarrow^2 q_{10}\ e_{3\text{-}2}\ q_{11}\ e_{3\text{-}9}\ \uparrow^1$; $\downarrow^3 q_5\ e_{3\text{-}5}\ q_6\ e_{3\text{-}6}\ q_7\ e_{3\text{-}7}\ q_8\ e_{3\text{-}5}\ q_9\ e_{3\text{-}3}\ \uparrow^4$.

Легко прослеживается связь строки ЛСПС и пути на графе: путь $q_1 \rightarrow q_4$ – это первая строка, заканчивающаяся фигурными скобками; остальные фрагменты графа представлены строками, стрелки которых указывают связь с первой строкой.

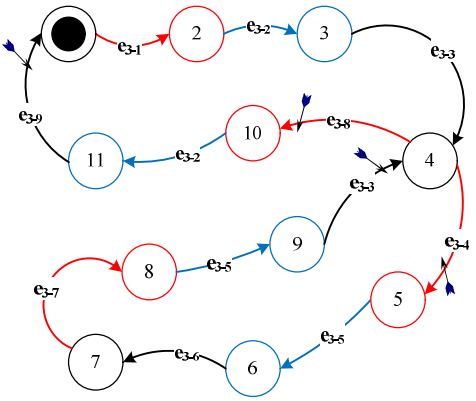

Рис. 1. Граф переходов автомата G^3

В работе использованы общепринятые обозначения для сетей Петри (см., например, [18], которые вполне согласуются с классической нотацией из [25]). Сеть Петри (СП) – это пара (S, μ_0), где S – некоторая сеть, а μ_0 – некоторая начальная разметка сети S. Сетью называется тройка $S = (P, T, F)$, где P и T – непустые непересекающиеся конечные множества, называемые местами и переходами; $F \subseteq (P \times T) \bigcup (T \times P)$ – бинарное отношение инцидентности, на основе которого вводится функция инцидентности $\overline{F} : (P \times T) \bigcup (T \times P) \rightarrow N$, где N – множество натуральных чисел. F для любого элемента $x \in P \bigcup T$ определя-

ет множества его входных и выходных элементов: ${}^{\bullet}x = \{y \mid yFx\}$ и $x^{\bullet} = \{y \mid xFy\}$. Разметкой (маркировкой) сети $S = (P, T, F)$ называется функция μ: $P \to N$. Если $\mu(p) = k \in \mathrm{N}$, то говорят, что в месте $p \in P$ имеется k фишек.

Переход $t \in T$ допустим на разметке μ СП $(S, \mu_0) \Leftrightarrow \forall p \in {}^{\bullet}t : \mu(p) \geq \overline{F}(p,t)$. Сеть в матричной форме задается матрицами W размерности $(n \times m)$ – по числу позиций и переходов. При этом входная $\overline{F}(t_j, p_i)$ и выходная $\overline{F}(p_j, t_i)$ функции переходов представляются матрицами W^+ и W^- соответственно. Матрица инцидентности $W = W^+ - W^-$ – целочисленная матрица $w_{ij} = \overline{F}(t_j, p_i) - \overline{F}(p_i, t_j)$, элементы которой задают то, какие изменения вносит в сеть срабатывание перехода t_j. Пусть $y(t_i)$ – вектор размерности m, у которого в позиции $y(i) = 1$, а в остальных 0. Тогда выражение $\mu = \mu' + W \cdot y^{\mathrm{T}}(t_i)$ в матричной форме представляет изменения маркировки сети, вызванные срабатыванием перехода t_1. По последовательности срабатывания переходов $s := t_1, t_2, \ldots, t_k$ построим вектор (размерности m) срабатывания (вектор Париха) y. Позиции y представляют число срабатываний каждого перехода в последовательности s. Тогда в векторной форме маркировка μ_k, достижимая после s, выглядит следующим образом: $\mu_k = \mu_0 + W \cdot y^{\mathrm{T}}$. При моделировании ДСС переходы взвешиваются именами *событий*.

P-инвариантом (*инвариантом позиций*) называют целочисленный вектор $X = \{x_i\}$, $i = 1, 2, \ldots, n$, являющийся решением линейной системы: $X \cdot W = 0$. *T*-инвариантом (*инвариантом переходов*) называется целочисленный вектор $Y = \{y_i\}$, $i = 1, 2, \ldots, m$, если он является решением линейной системы: $W \cdot Y^{\mathrm{T}} = 0$.

3. Методика сквозного проектирования супервизорного управления

На первом этапе проектирования супервизора для ДСС в работе предлагается провести СДСС-моделирование объекта управления и спецификации его требуемого поведения.

Мотивацию этого решения автор видит в следующем. Формализация структуры и поведения на первых этапах моделирования сложного объекта обычно основывается на сложившейся практике (традиции, опыте) разработчиков (проектантов) объекта и технологии. Базовым понятием технологических данных является технологическая операция, базовой компонентой структуры чаще всего является исполнительный механизм (актуатор), а различного рода диаграммы, циклограммы и/или блок-схемы описывают требуемое (допустимое, востребованное) поведение. Технологическая операция характеризуется событием (командой на исполнение), изменением состояния объекта и соответствующими откликами (реагированием). При моделировании ДСС на сетях Петри, характеризующихся очень бедным набором базовых средств (позиция и переход), возникают объективные методологические трудности. Одна из них: как технологическую операцию представить (моделировать) позициями и переходами. На наш взгляд, целесообразно на этапе первичной формализации использовать модель конечный автомат, поскольку состояния, условия (события) их завершения и функции переходов и выходов достаточно прозрачно корреспондируются с упомянутыми выше традиционными для технологов инструментами (диаграммы, циклограммы и/или блок-схемы).

3.1. МОДЕЛИРОВАНИЕ СТРУКТУРЫ ОБЪЕКТА И ФУНКЦИОНАЛЬНОСТИ КОМПОНЕНТ КАК ДСС, ПРЕДСТАВЕННОЙ НАБОРОМ КОНЕЧНЫХ АВТОМАТОВ

Структуризация объекта начинается с выделения всех актуаторов (приводов и исполнительных механизмов); пусть это будет $G = \langle G^1, G^2, ..., G^n \rangle$. Для каждого G^i определяются: мно-

жество событий, являющихся командами E_c^i, множество событий E_w^i, характеризующих траекторию движения этого конкретного механизма (множество реакций на E_c^i) и, наконец, определяется множество условий, внешних относительно $G^i - E_{uc}$. Общий алфавит для $G^i - E^i = \{E_w^i \cup E_c^i \cup E_{uc}^i\}$. Взаимосвязь команд ($E_c^i$), ожидаемых реакций и внешних условий (если это требуется) на практике представляется в виде диаграмм (циклограмм), что достаточно прозрачно формализуется конечным автоматом $G^i = <Q^i, E^i, \delta_i, \Gamma^i, Q_m^i, q_0^i>$. Здесь и далее такие автоматы будем называть компонентными конечными автоматами (ККА). В итоге анализа объекта определяются: состав СДСС $G = \langle G^1, G^2, ..., G^n \rangle$, множества E^i событий, каждое из которых структурируется на $E^i = \{E^i_w \cup E^i_c \cup E^i_{uc}\}$, и множество E_{uc} – общих неуправляемых событий. Пример графа переходов ККА модели такого механизма представлен на рис. 2а).

Следующим этапом построения СДСС является определение востребованного поведения набора компонент. Поведение представляется как последовательность команд из $E_c = \bigcup_{i=1}^{n} E_c^i$, заданной строками в языке $K \in E_d^*$. При этом в спецификацию входят только строки, соответствующие множеству путей на графе переходов конечного автомата $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m^h, q_0)$. В примере на рис. 2 представлен ККА первого механизма, а второй в точности до обозначений событий (индекс начинается с 2) опущен. Спецификация их востребованного поведения представлена на рис. 2б).

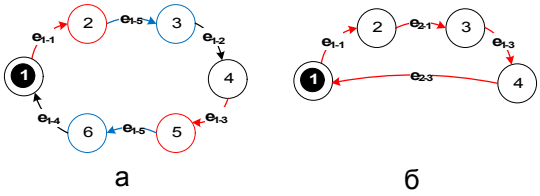

Рис 2. а) ККА – модель механизма G^1;
б) автомат спецификации H

Завершающим этапом СДСС-моделирования является проверка управляемости на основе алгоритма эксперимента $\Re(s)$, который является основным инструментом при изучении совместного поведения G и H. Результатом эксперимента $\Re$ над строками $s \in K$ (все строки допустимы для начального состояния q_0) может быть $\Re(s) = True \mid False$. Если после всех действий над символами s все компоненты G успешно осуществили свои переходы, то $\Re(s) = True$; если во время эксперимента новый символ $s(i)$ не «сработал» в соответствующей компоненте, тогда $\Re(s) = False$. В последнем случае подразумевается, что H не согласован с G и спецификация требуемого поведения (автомат $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m{}^h, q_0)$) должна быть изменена. В завершение анализа определим $Q = \{Q_1, Q_2, \ldots, Q_r\}$ – множество достижимых векторов-состояний компонент $G = \left\langle G^1, G^2, ..., G^n \right\rangle$, где $Q_j := \left\langle q^1_{j1}, q^2_{j2}, ..., q^n_{jn} \right\rangle$ и каждое $Q_j \in Q$ соотносится с $q_j \in Q^h$.

3.2. СПЕЦИФИЦИРОВАНИЕ КОМПОНЕНТНЫХ КОНЕЧНЫХ АВТОМАТОВ ЛОГИЧЕСКИМИ СХЕМАМИ ПОСЛЕДОВАТЕЛЬНОСТИ СТРОК

Для систем промышленной автоматизации довольно типичны простые механизмы, функциональность которых заключается в выполнении по циклу операций. В соответствии с правилами, изложенными в разделе 2, для графов переходов моделей таких механизмов формируем ЛСПС. Для G^1(см. рис. 2а) ЛСПС: $\downarrow^1 q_1\ e_{1\text{-}1}\ q_2\ e_{1\text{-}5}\ q_3\ e_{1\text{-}2}\ q_4\ e_{1\text{-}3}\ q_5\ e_{1\text{-}5}\ q_6\ e_{1\text{-}4}\ \uparrow^1$.

Построение ЛСПС по графу спецификации аналогично. Иллюстрируем это примером двух механизмов G^1 и G^2. Пусть G^2 устроен аналогично G^1 и только индексы событий имеют префикс 2, тогда ЛСПС G^2 имеет вид: $\downarrow^1 q_1\ e_{2\text{-}1}\ q_2\ e_{2\text{-}5}\ q_3\ e_{2\text{-}2}\ q_4\ e_{2\text{-}3}\ q_5\ e_{2\text{-}5}\ q_6\ e_{2\text{-}4}\ \uparrow^1$. Пусть требуется, чтобы циклически последовательно выполнялись операции механизмов G^1 и G^2: вначале первые – $e_{1\text{-}1}$, $e_{2\text{-}1}$, а затем вторые – $e_{1\text{-}3}$, $e_{2\text{-}3}$. Граф переходов автомата H в СДСС-модели имеет вид, представленный на

рис. 2б). Напомним, что в автомате H не используются ожидаемые события. Тогда ЛСПС автомата $H - LS^H$ имеет следующий вид: $\downarrow^1 q_1\ e_{1\text{-}1}\ q_2\ e_{2\text{-}1}\ q_3\ e_{1\text{-}3}\ q_4\ e_{2\text{-}3} \uparrow^1$.

3.3. СИНТЕЗ СЕТИ ПЕТРИ, МОДЕЛИРУЮЩЕЙ ТЕХНОЛОГИЧЕСКИЙ ОБЪЕКТ УПРАВЛЕНИЯ

Моделирование объекта как СДСС включает все события, представляющие детально его поведение. Каждая операция механизма представляется строкой, начинающейся с управляемого события – команды, и последовательности всех последующих ожидаемых событий. Так, например, для G^1 операция «зажим детали» представляется строкой: $e_{1\text{-}1}\ q_2\ e_{1\text{-}5}\ q_3\ e_{1\text{-}2}\ q_4$. Вместе с тем для моделирования этой операции сетью Петри достаточно после события – команды $e_{1\text{-}1}$ перейти в позицию выполнения операции, а по событию $e_{1\text{-}2}$ перейти в позицию, соответствующую ее завершению. Это преобразование сформулируем как правило редукции строк ЛСПС.

3.3.1. ПРАВИЛО РЕДУКЦИИ СТРОК ЛСПС

Подстрока из 4-х символов типа: <*состояние*> i_1, <*событие*> j_1, <*состояние*> i_2, <*событие*> j_2, в которой все события относятся к типу ожидаемых, редуцируется в подстроку: <*состояние*> i_2, <*событие*> j_2.

Применим правило редукции к ЛСПС G^1:

$\downarrow^1 q_1\ e_{1\text{-}1}\ q_2\ e_{1\text{-}5}\ q_3\ e_{1\text{-}2}\ q_4\ e_{1\text{-}3}\ q_5\ e_{1\text{-}5}\ q_6\ e_{1\text{-}4} \uparrow^1$.

Подстрока из 4 символов, например

$q_2\ e_{1\text{-}5}\ q_3\ e_{1\text{-}2} \Rightarrow \sout{q_2\ e_{1\text{-}5}}\ q_3\ e_{1\text{-}2} \Rightarrow q_3\ e_{1\text{-}2}$.

В соответствии с правилом редукции строка ЛСПС G^1 последовательно преобразуется

$\downarrow^1 q_1\ e_{1\text{-}1}\ q_2\ e_{1\text{-}5}\ q_3\ e_{1\text{-}2}\ q_4\ e_{1\text{-}3}\ q_5\ e_{1\text{-}5}\ q_6\ e_{1\text{-}4} \uparrow^1 \Rightarrow$

$\downarrow^1 q_1\ e_{1\text{-}1}\ q_3\ e_{1\text{-}2}\ q_4\ e_{1\text{-}3}\ q_6\ e_{1\text{-}4} \uparrow^1$.

3.3.2. КОНСТРУИРОВАНИЕ СЕТИ ПЕТРИ

После редукции ЛСПС синтез СП заключается в замене символов состояний символами позиций, сопоставлении событиям переходов и в построении по строкам ЛСПС (в соответствии со следованием в них символов) сети Петри. Введем ряд обозначений. Далее ЛСПС автомата G^i , для краткости, будем обозначать LS^i. Если подстрока h встречается в ЛСПС LS^i, то это будем обозначать $h \leftarrow\!\shortmid LS^i$.

Задача этапа конструирование СП для механизмов состоит в преобразовании LS^i.

$G = \langle G^1, G^2, ..., G^n \rangle \Rightarrow S_p = \bigcup_{i=1}^{n} S^i$, при этом каждый $G^i = <Q^i, E^i, \delta_i, \Gamma^i, Q_m{}^i, q_0{}^i>$ преобразуется в $S^i = \{P^i, T^i, F^i, \mu_0\}$ следующим образом: $P^i = \{p_k \,|\, q_k \leftarrow\!\shortmid LS^i\}$; $T^i = \{t_{k-l} \,|\, e_{k-l} \leftarrow\!\shortmid LS^i\}$.

Структура переходов определяется через входные и выходные функции переходов и соответствующие подстроки LS^i: ${}^\bullet t_{k-l} = \{p_j \,|\, q_j e_{k-l} \leftarrow\!\shortmid LS^i\}$; $t_{k-l}^\bullet = \{p_j \,|\, e_{k-l} q_j \leftarrow\!\shortmid LS^i\}$. Вектор начальной маркировки $\mu_0{}^i$ содержит метку в позиции, соответствующей $q_0{}^i$. Для механизмов G^1 и G^2 сеть имеет вид, представленный на рис. 3.

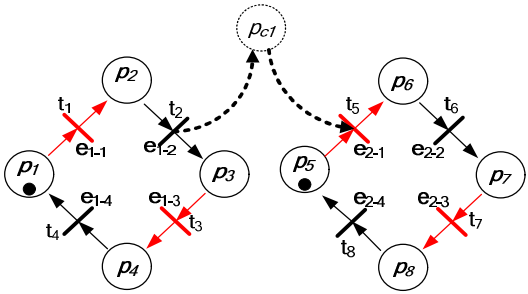

Рис. 3 Сеть Петри процесса для G^1 и G^2

Конструируем по автомату H соответствующую ЛСПС LS^H и сделаем ее расширение: каждый переход t_i, соответствующий

событию – команде, заменим парой $t_i t_k$, где t_k переход, завершающий операцию t_i в соответствующем S^i. ЛСПС автомата H – LS^H имеет следующий вид: $\downarrow^1 q_1\ e_{1\text{-}1}\ e_{1\text{-}2}\ q_2\ e_{2\text{-}1}\ e_{2\text{-}2}\ q_3\ e_{1\text{-}3}\ e_{1\text{-}4}\ q_4\ e_{2\text{-}3}\ e_{2\text{-}4} \uparrow^1$.

3.4. СИНТЕЗ УПРАВЛЯЮЩЕЙ СЕТИ ПО SP И СПЕЦИФИКАЦИИ ПОСЛЕДОВАТЕЛЬНОСТЕЙ КОМАНД

Основная идея предлагаемого нами метода основывается на следующем свойстве циклических реактивных систем с форсируемыми событиями: каждая следующая операция объекта выполняются по событию завершения предшествующей операции (подобно падению выставленных вертикально костяшек домино) и/или внешнему событию (например, при пуске или выборе). В этом случае задача супервизорного управления – организовать передачу активности (управления) от механизма к механизму по завершению их операций. Метод синтеза управляющей сети Петри мы назвали методом домино.

Введем необходимые понятия и отметим ряд свойств сети процесса, которые в принципе доказуемы, но ограниченный объем статьи не позволяет это изложить. Исключим из строк, входящих в LS^H, символы состояний. Далее в тексте все события e_{i-j} заменим именами переходов t_k и перенумеруем сплошной нумерацией по мере их следование в S_p, например, как это показано на рис 3. Тогда строка LS^H примет вид $\downarrow^1 t_1 t_2 t_5 t_6 t_3 t_4 t_7 t_8 \uparrow^1$.

Несвязанные пары переходов. Переход $t_j\ t_i$ сети S_p называется несвязанной парой (н-парой), если $t_j\ t_i \not\dashv LS^H$ и в сети S_p не существует позиции p_k такой, что $t_j \in {}^\bullet p_k$, а $t_i \in p_k^\bullet$.

***Свойство* 1**. При преобразовании сети S_p в S_p'. путем расширения S_p звеном $t_j \to p_c \to t_i$ (далее просто $t_j\, p_c\, t_i$) P-инварианты сети S_p сохраняются в сети S_p' (преобразование иллюстрирует звено на рис. 3, выделенное пунктиром). Однако множество достижимых состояний, несомненно, изменится. Более того, ограниченная сеть S_p преобразуется в неограничен-

ную S_p'., поскольку в позиции p_c возможен неограниченный рост числа меток при бесконечном цикле в первой сети.

***Свойство* 2**. Для всякой н-пары $t_j\ t_i$ скалярное произведение векторов – столбцов $(W(t_i), W(t_j)) = 0$. Следует из факта отсутствия связующих их позиции.

***Свойство* 3.** Пусть расширение сети S_p осуществляется множеством звеньев, сконструированных для всех н-пар из последовательности срабатывания u, соответствующей LS^H. При этом вектор Парика $Y(LS^H)$, соответствующий u, является T-инвариантом сети S_p, поскольку LS^H задает циклический процесс, а $G = (G^1, G^2, \ldots, G^n)$ и $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m{}^h, q_0)$ согласованы. Тогда существует изоморфное отображение $Q = \{Q_1, Q_2, \ldots, Q_r\}$ – множества достижимых векторов-состояний компонент $G = \left\langle G^1, G^2, ..., G^n \right\rangle$ (где $Q_j := \left\langle q_{j1}^1, q_{j2}^2, ..., q_{jn}^n \right\rangle$), на M' – множество достижимых маркировок сети S_p'. Кроме того, P-инварианты сети S_p сохраняются в S_p' и $Y(LS^H)$ является T-инварианты сети S_p'.

Из сформулированных свойств следует, что если для всех возможных последовательностей строк, задаваемых, например, ЛСПС, выявить все н-пары $t_j\ t_i$ и каждую пару в сети процесса обеспечить звеном $t_j\ p_c\ t_i$ передачи управления (активности), то, во-первых, все структурные свойства исходной сети процесса сохраняются; во-вторых, все требуемые последовательности операций, определяемые спецификацией, будут выполняться.

В этом разделе мы ограничимся разработкой ядра метода ограничивающего LS^H бесповторной[4] ЛСПС, содержащей одну последовательность. Это ограничение далее будет снято.

Опишем процедуру проектирования управляющей сети S_c из СПУ $t_j\ p_c\ t_i$ для ЛСПС, содержащей одну бесповторную последовательность срабатывания переходов (управляющую сеть S_c иногда называют сеть контроллера).

---

[4] *В бесповторной последовательности срабатывания каждый переход упоминается не более 1 раза.*

Процедура построения управляющей сети S_c.

Обозначения, используемые в тексте процедуры: ***Sintez*** – имя процедуры; ***W***, ***Pc***, ***n***, ***u***, ***r*** – формальные параметры**;** $u := t_1\ t_2\ t_3\ \ldots t_{r-1}\ t_r$ – строка ЛСПС, обрабатываемая процедурой (в u стрелочки заменены именами переходов, на которые эти стрелочки указывают в ЛСПС); r – длина строки u; $W(n \times m)$ – матрица инцидентности сети процесса S_p; $W(k, \sim) = 0$ означает присвоение всем элементам строки k матрицы W значение 0; $u(i)$ i-ый символ строки u. $W(u(i))$, $W(u(i+1))$ – векторы-столбцы матрицы W, $SProdct(W(u(i)), W(u(i+1)))$ – скалярное произведение векторов. Текст процедуры:

***Sintez* (*W*, *Pc*, *n*, *u*, *r*)**
***i* = 1**
While i* ≤ (*r* – 1) *do
Begin
***If Sprodct* (*W*(*u*(*i*)), *W*(*u*(*i* + 1))) ≠ 0 *go to M*1;**
***n* = *n* + 1;**
***W*(*n*, ~) = 0;**
***W*(*n*, *u*(*i*)) = +1;**
***W*(*n*, *u*(*i* + 1)) = -1;**
***Pc* = {*Pc*} ∪ {p_n};**
***M*1: *i* = *i* + 1**
End.

Применим процедуру ***Sintez*** к сети S_p (см. рис. 3) для двух механизмов G^1 и G^2. Матрица инцидентности W этой сети представлена в таблице 1. В матрице W имена событий заменены именами соответствующих переходов следующим образом: $(e_{1\text{-}1} \to t_1)$ …$(e_{2\text{-}4} \to t_8)$. Обрабатываемая строка

$u := t_1\ t_2\ t_5\ t_6\ t_3\ t_4\ t_7\ t_8\ t_1$; $r = 9$.

Результатом работы процедуры ***Sintez*** является построение сети S_c из четырех управляющих позиций (p_{c1}, …, p_{c4}). Объединенная сеть S_p и S_c представлена на рис. 4.

Вычисление μ_0. Вектор начальной маркировки $\mu_0^{\mathrm{T}} = [\mu_{0P}^{\mathrm{T}}\ \mu_{0C}^{\mathrm{T}}]$ состоит из начальной разметки μ_{0P}^{T} сети процесса S_p и начальной разметки μ_{0C}^{T} сети контроллера S_c.

Первая часть μ_{0P}^{T} определяется при построении S_p и соответствует начальным состояниям ЛСПС (в позициях вектора μ_{0P}^{T}, соответствующих начальным состояниям в определении ЛСПС присваивается 1, а остальным 0).

Правило для начальной разметки μ_{0C}^{T}. Поскольку вектор Y является T-инвариантом (задает цикл), то если из него исключить срабатывание последних в цикле переходов, то матричная операция W с модифицированным Y' определит начальную разметку для S_c.

Таблица 1. *Матрица инцидентности W для СП механизмов* G^1 *и* G^2

	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8
P_1	–1	0	0	1	0	0	0	0
P_2	1	–1	0	0	0	0	0	0
P_3	0	1	–1	0	0	0	0	0
P_4	0	0	1	–1	0	0	0	0
P_5	0	0	0	0	–1	0	0	1
P_6	0	0	0	0	1	–1	0	0
P_7	0	0	0	0	0	1	–1	0
P_8	0	0	0	0	0	0	1	–1
Pc_1	0	+1	0	0	–1	0	0	0
Pc_2	0	0	–1	0	0	+1	0	0
Pc_3	0	0	0	+1	0	0	–1	0
Pc_4	–1	0	0	0	0	0	0	+1

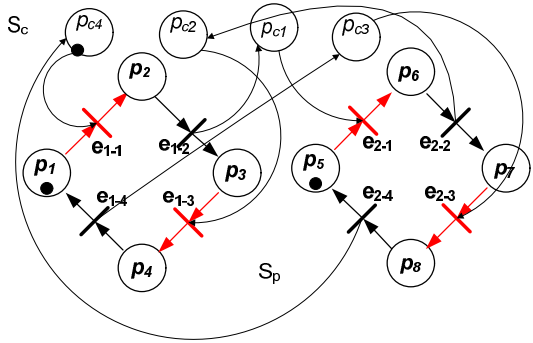

Рис. 4. Сеть Петри с управляющими позициям, реализующими управление, так, что выполняется последовательность $u := t_1\ t_2\ t_5\ t_6\ t_3\ t_4\ t_7\ t_8\ t_1$

3.5. УЛУЧШЕНИЕ SC ПУТЕМ МИНИМИЗАЦИИ КАНАЛОВ УПРАВЛЕНИЯ

Процедура ***Sintez*** вводит позицию, передающую управление, для каждой н-пары подобно каналу связи. Эти каналы «работают» только в момент активности соответствующей н-пары и после передачи управления в следующие переходы не участвуют в процессе активизации других переходов, пока в следующем цикле они не будут вновь востребованы. Из чего следует, что формирование для каждой н-пары нового канала в сети во многих случаях избыточно. Прозрачно определяются отношение совместимости каналов (по отсутствию пересечения предшественников и последователей) и соответствующие правила. Эти правила подобны правилам совместимости состояний для конечного автомата. Применив эти правила и проделав традиционные преобразования, для нашего примера удалось сократить число позиций в два раза. Результат представлен на рис 5.

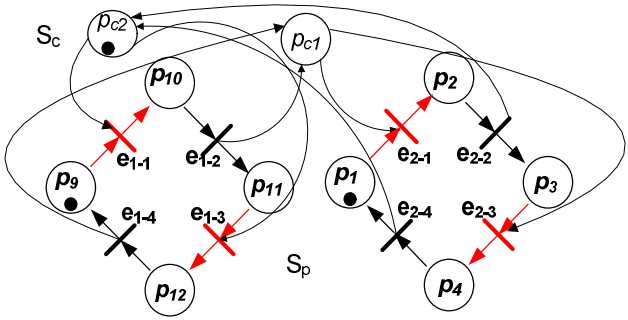

Рис. 5. Преобразованная сеть Петри, выполняющая последовательность $u := t_1\ t_2\ t_5\ t_6\ t_3\ t_4\ t_7\ t_8\ t_1$

3.6. СИНТЕЗ УПРАВЛЯЮЩЕЙ СЕТИ ПО ЛСПС, ПОСТРОЕННОЙ ПО СПЕЦИФИКАЦИИ СДСС БЕЗ ОГРАНИЧЕНИЙ НА БЕСПОВТОРНОСТЬ И НАЛИЧИЕ ВЕТВЛЕНИЙ

В настоящем разделе мы ограничимся изложением основных положений развития процедуры синтеза S_c в направлении отказа от ограничений на бесповторность и количество строк в *LS*.

В спецификации требуемого поведения в общем случае допустимы многократные срабатывания механизмов и многовариантное поведение, регулируемое предшествующими срабатываниями механизмов в уже отработанных последовательностях или внешними (неуправляемыми) событиями. Последнее представляется ветвлениями в графе переходов автомата $H = (Q^h, E_d, \delta_h, \Gamma^h, Q_m{}^h, q_0)$, которые в ЛСПС представляются ветвителями.

Многократность и вариантность срабатывания механизмов приводит к многообразию в сочетании событий в н-парах (срабатывание перехода будем называть событием). Возможны различные н-пары, в которых совпадают 1-е или вторые 2-е события. Рассмотрим, как это отразится на структуре звеньев передачи управления.

Замечание 1. Совпадение первого события в различных н-парах. Наличие в *LS* двух или более н-пар с совпадающими первыми символами означает, что одно и то же событие, например t_i, в различных частях последовательности – спецификации – влечет различные последствия: $t_i - t_k$ и $t_i - t_r$. Следовательно, необходимо в структуре звена передачи управления привлечь дополнительную информацию (другие события или их последовательности, которые различают н-пары $t_i - t_k$ и $t_i - t_r$).

Замечание 2. Совпадение второго события. Наличие двух или более н-пар типа $t_i - t_k$, $t_j - t_k$, …, $t_n - t_k$ означает, что переход t_k срабатывает всякий раз, когда срабатывает один из переходов t_i, t_j, …, t_n. Таким образом, в структуре звена передачи управления должна производиться сборка по функции «ИЛИ». Для этого в процедуре конструирования звена передачи управления всякая н-пара, например $t_i - t_k$, будет анализироваться на возможность «повтора 2-го символа». При обнаружении такой н-пары, например $t_j - t_k$, в звене вводится позиция, в которой осуществляется сборка передачи управления в общий для всех н-пар переход t_k, в который из позиции-сборки вводится ребро $p_{or} - t_k$.

Из замечаний 1 и 2 следует, что локальный анализа н-пары недостаточен для выбора типа входной и выходной частей звена передачи управления. При анализе н-пары необходима информация о наличии совпадений по первому и второму событию с другими н-парами в *LS*. Выполним просмотр всех последовательностей, входящих в *LS* и снабдим каждую н-пару соответствующим признаком.

Отметим одну из особенностей процедуры ***Sintez***: позиция сети S_c вводится, как только обнаруживается н-пара, вне зависимости от предшествующих и последующих событий в строке. Следовательно, можно выполнять эту процедуру и по множеству н-пар. Метод синтеза управляющей сети S_c по ЛСПС общего вида формулируется как циклическая процедура перебора строк из *LS*, анализа всех н-пар в каждой строке, и формирования для каждой группы н-пар, с учетом замечаний 1 и 2 структуры звена

передачи управления. Строки нам важны по двум причинам: во-первых, как источники н-пар, во-вторых, для оценки сложности самой процедуры.

Легко видеть справедливость следующего утверждения.

Утверждение. Сложность выполнения процедуры ***Sintez*** пропорциональна суммарной длине строк в *LS*.

4. Заключение

В работе для интеграции систем автоматизации, работающих в реальном времени, разработана методология синтеза управляющей сети Петри, базирующаяся на важнейшем концепте сетецентрического управления: *неизменность функциональности автономных компонент и выполнение на уровне сетевой интеграции обработки информации в интересах потребителей*.

Методология основана на использовании структурированной дискретно-событийной системы (СДСС) для анализа функциональности и согласованности модели. Разработан метод синтеза по СДСС-модели сети Петри S_p, моделирующей процессы в объекте, разработана техника анализа S_p и метод синтеза управляющей сети супервизора – S_c, обеспечивающей совместно с сетью S_p выполнение исходных спецификаций. Метод синтеза управляющей сети S_c основан на анализе ЛСПС – компактной модели последовательности срабатывания переходов в сети процесса. Предложенная модель СДСС и процедуры работы с ней максимально учитывают особенности систем автоматизации, работающих в реальном масштабе времени, отмеченные во введении. Есть основание полагать (*утверждение* в разделе 3.6), что в предложенном методе, основанном на линейном просмотре всех простых строк из ЛСПС, удается избежать «взрыва состояний» при синтезе супервизора.

Важным свойством управляющей сети Петри, синтезированной по предложенной методике, является то, что в ее структуре сохранена S_p – СП-модель актуаторов (исполнительных механизмов нижнего уровня). Сеть Петри S_c, управляющая

интеграцией компонент нижнего уровня (супервизор), синтезирована как отдельная компонента, взаимодействующая с сетями нижнего уровня. Важно отметить, что сети S_c нет в структуре исходных данных. Напротив – эта сеть есть результат синтеза обратных связей в S_p на основе анализа возможностей модели процессов в объекте – S_p и требований спецификации *LS*. Таким образом, предложенный метод синтезирует двухуровневую управляющую систему, отвечающую требования сетецентрической интеграции, сформулированным во введении.

Предлагаемая методика изложена применительно к механической системе. Однако нет никаких ограничений на применение сформулированных результатов к синтезу супервизорного управления бизнес-процессами в организационных системах. При этом строки из команд (директив на выполнение вариантов бизнес-процессов) языка супервизора K можно интерпретировать как различные варианты достижения цели функционирования. Синтезированный супервизор – сценарий заданный сетью Петри; изменение сценария – смена матрицы, а в реальной организационной системе смена текущего плана мероприятий.

Литература

1. АМБАРЦУМЯН А.А. *Супервизорное управление в дискретно событийных системах* // Автоматика и телемеханика. – 2009. – №8. – С. 156–166.
2. АМБАРЦУМЯН А.А. ПРАНГИШВИЛИ И.В., ПОЛЕТЫКИН А.Г. *Анализ состояния и предложения по повышению уровня автоматизации энергетических объектов* // Проблемы управления. – 2003. – №2. – С. 37–57.
3. АМБАРЦУМЯН А.А., ТОМИЛИН Е.Е. *Метод прямого синтеза супервизора для структурированной дискретно-динамической системы* // Автоматика и телемеханика. – 2010. – №8. – С. 168–188.
4. АМБАРЦУМЯН А.А. *Эквивалентные преобразования управляющих автоматов, генерируемых по описанию пове-*

дения объекта управления // Тезисы докладов Первой Международной конференции молодых ученых «Проблемы проектирования и применения дискретных систем в управлении», Минск: ИТК АН БССР, 1977. – С. 77–78.

5. АМБАРЦУМЯН А.А., ГРИГОРЯН А.К. *О соотношениях управляющих автоматов, порождаемых по их описанию на первичном языке* // Автоматика и телемеханика. – 1978. – №12. – С. 130–138.
6. ЗАКРЕВСКИЙ А.Д., ПОТТОСИН Ю.В., ЧЕРЕМИСИНОВА Л.Д. *Логические основы проектирования дискретных устройств*. – М.: Физматлит, 2007 – 588 с.
7. ЗАЦАРИННЫЙ А.А., ПРИСЯЖНЮК С.П., ИОНЕНКОВ Ю.С. *Тенденции развития современных инфокоммуникационных технологий с учетом концепции сетецентрических войн* // Информация и космос. – 2006. – №4. – С. 85–93. – URL: http://elibrary.ru/contents.asp?issueid=644974.
8. ИВАНОВ Н.Н., МИХАЙЛОВ Г.И., РУДНЕВ В.В. ТАЛЬ А.А. *Конечные автоматы: эквивалентность и поведение*. – М.: Наука, 1984. – 192 с.
9. ИВЛЕВ А.А. *Основы теории Бойда. Направления развития, применения и реализации*. Москва, 2008. – 64 с. – URL: http://old.vko.ru/pdf/2008/library/08_05_23_02.pdf.
10. ЛАЗАРЕВ В.Г., ПИЙЛЬ Е.И. *Синтез управляющих автоматов*. – М.: Энергоатомиздат, 1989.
11. РУДНЕВ В.В. *Конечный автомат как объект управления* // Автоматика и телемеханика. – 1978 – №9. – С. 126–135.
12. ТАЛЬ А.А., ЮДИЦКИЙ С.А. *Иерархия и параллелизм в сетях Петри* // Автоматика и телемеханика – 1982. – I: №7. – С. 121-139; II: №8, – С. 111–128.
13. ЮДИЦКИЙ С.А. *К вопросу описания и синтеза дискретных систем промышленной автоматики* // Техническая кибернетика. – 1976. – №1. – С. 131–141.
14. AKESSON K., FLORDAL H., FABIAN M. *Exploiting modularity for synthesis and verification of supervisors* // Proc. of the IFAC World Congress, Barcelona, Spain, July. 2002. – P. 1444–1449.

15. AMBARTSUMYAN A.A., KAZANSKY D.L. *Complex automation of technological processes with involving event model in feedback control scheme* // Proceedings of 17th IFAC World Congress, Seoul, Korea, 2008. – P. 28–33.
16. AMBARTSUMYAN A.A., KAZANSKY D.L. *An approach to technological process control systems based on model with technological coalitions* // Proceedings of 12th International conference on systems engineering (ICSEng08), Las Vegas, USA, 2008. – P. 219–224.
17. BARANOV S. *Logic and System Design of Digital Systems*. – Toronto: Published jointly Tallinn University of Technology Press and SIB Publishers, 2008. – P. 266.
18. CASSANDRAS C.G., LAFORTUNE S. *Introduction to discrete event systems*. – Dordrecht: Kluwer Academic Publishers, 2008. – P. 848.
19. DE QUEIROZ M.H., CURY J.E.R. *Modular supervisory control of large scale discrete event systems* // Discrete Event Systems: Analysis and Control, Proc. WODES'00, 2000. – P. 103–110.
20. DIDEBAN A., ALLA H. *Reduction of Constraints for Controller Synthesis based on Safe Petri Nets* // Automatica. – 2008. – No. 44(7). – P. 1697–1706.
21. GAUDIN B., MARCHAND H. *Modular supervisory control of asynchronous and hierarchical finite state machines* // In European Control Conference, Cambridge, 2003. – P. 133–138.
22. GOLASZESKI C.H., RAMADGE P.J. *Control of discrete event processes with forced events* // Proc. 28th Conference on Decision and Control, Los Angeles, 1987. – P. 247–251.
23. HOLLOWAY L.E., KROGH B.H. *Synthesis of feedback logic for a class of controlled Petri nets* // IEEE Trans. Autom. Control. – 1990. – P. 514–523.
24. RAMADGE J.G., WONHAM W.M. *Supervisory control of a class of discrete event processes* // SIAM Journal of Control and Optimization. – 1987. – No. 25. – P. 206–230.
25. PETERSON J.L. *Petri Net theory and the modeling of systems*. – Englewood Cliffs, N.J: Prentice-Hall, Inc., 1981.

26. WONHAM W.M., RAMADGE J.G. *Modular supervisory control of discrete event systems* // Math Control Signals and Systems. – 1988. – No. 1. – P. 13–30.
27. YAMALIDOU K., MOODY J.O., LEMMON M.D., ANTSAKLIS P.J. *Feedback control of Petri nets based on place invariants* // IEEE Transaction on Robotics and Automation. – 1996. – No. 32(1). – P. 15–28.
28. YOO T.-S., LAFORTUNE S. *A general architecture for decentralized supervisory control of discrete event systems* // Discrete Event Dynamic Systems: Theory & Applications. – 2002. – No. 12(3). – P. 335–337.

NETWORK-CENTRIC CONTROL ON PETRI NETS FOR STRUCTURED DISCRETE EVENT SYSTEM

Alexander A. Ambartsumyan, Institute of Control Sciences, Russian Academy of Science(s), Moscow, Russia; (tel.: +7-495-334-87-89, e-mail: ambar@ipu.ru)

Abstract: *The paper deals with the methodology of control Petri net synthesis developed for real-time automation systems. The methodology is based on the model called a structured discrete event system* (*SDES*) *applied to analysis functionality and consistency. In the paper, SDES structure is defined, the technique of SDES controllability analysis is proposed. Developed are the method* (*based on SDES model*) *of Petri net synthesis for process modeling, the technique of process net analysis and the synthetic method of supervisor control net providing, jointly with the process net, the fulfillment of initial specifications.*

Keywords: logical control, control system, network-centric system, supervisor, discrete event systems, event model, Petri net.

Статья представлена к публикации членом редакционной коллегии М. В. Губко