УДК 658.512

Н.Н. Войт

ДОКАЗАТЕЛЬСТВО КОРРЕКТНОСТИ ГРАФ-СХЕМ ПОТОКОВ ПРОЕКТНЫХ РАБОТ В АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ¹

Войт Николай Николаевич, кандидат технических наук, доцент, окончил факультет информационных систем и технологий Ульяновского государственного технического университета, доцент кафедры «Вычислительная техника» УлГТУ. Имеет статьи в области интеллектуальных САПР, Case-, Cals-технологий. Область научных интересов: автоматизация проектирования потоков работ в базе графических языков, формальные языки, грамматики и автоматы, контроль и анализ бизнес-процессов, интеллектуальные системы обработки workflows, инженерия онтологии, автоматизированные среды обучения. [e-mail: n.voit@ulstu.ru].

Аннотация

Автор доказывает конструктивным подходом корректность граф-схем потоков проектных работ $G(AM)$, представленных в базах графических языков AM -модели. В работе определены типовые некорректные конструкции, приемы построения корректных граф-схем потоков проектных работ, даны определение, теорема о корректности граф-схем потоков проектных работ и алгоритмы преобразования граф-схем алгоритмов, граф-схемы потоков проектных работ.

Полученные результаты являются новыми знаниями в изучении корректности сложных технических систем. В практическом плане результаты позволят проектировщикам избежать дорогостоящих проектных ошибок в автоматизации проектирования сложных изделий на промышленном предприятии (в том числе крупном, где количество номенклатуры превышает 1,5 млн. позиций и количество разработчиков больше 1000 человек). Материал данной работы связан с авторскими публикациями в предыдущих журналах «Автоматизация процессов управления» № 3 (61) и № 4 (62) 2020 года.

Ключевые слова: корректность алгоритма, бизнес-процессы, графические языки, потоки проектных работ, диаграмматика.

doi: 10.35752/1991-2927-2021-2-64-23-33

EXAMPLE OF ANALYTICAL MODEL FOR ORGANIZATION IN DESIGN WORKFLOWS

Nikolai Nikolaevich Voit, Candidate of Sciences in Engineering; Associate Professor; graduated from the Faculty of Information Systems and Technologies of Ulyanovsk State Technical University; Associate Professor of the Department of Computer Engineering of UlSTU; an author of articles in the field of intelligent CAD, Case-, Cals-Technologies; research interests are in the field of workflow design automation in the basis of graphic languages, formal languages, grammars and automata, monitoring and analysis of business processes, intelligent workflow processing systems, ontology engineering, automated learning environments. e-mail: n.voit@ulstu.ru.

Abstract

The author uses a constructive approach to prove the correctness of the graph schemes of the $G(AM)$ project workflows presented in the bases of the graphical languages of the AM -model. The paper defines typical incorrect constructions, techniques for constructing correct graph schemes of design workflows, and provides a definition, a theorem on the correctness of graph schemes of design workflows, and algorithms for converting graph schemes of algorithms, graph schemes of design work flows.

The results obtained are new knowledge in the study of the correctness of complex technical systems. In practical terms, the results will allow designers to avoid costly design errors in automating the design of complex products at an industrial

¹ Исследование выполнено при финансовой поддержке РФФИ и Ульяновской области в рамках научного проекта № 18-47-730032.

enterprise (including a large one – the number of nomenclature exceeds 1.5 million items and the number of developers exceeds 1000 people). The material of this work is related to the author's publications in the previous issues of Automation of Control Processes Journal, no. 3(61) and no. 4(62), 2020.

Keywords: algorithm validation, business processes, graphical languages, design workflows, diagrammatic.

ВВЕДЕНИЕ

Настоящая работа является развитием темы по обработке потоков проектных работ (*workflow*) в САПР при проектировании сложных технических систем [1–3]. Фокус внимания сосредоточен на одном из важнейших свойств любой системы – корректности. Определение корректности *workflow* является важной задачей в теории управления бизнес-процессами предприятия. В области теоретической информатики свойство корректности является одним из фундаментальных при изучении алгоритмов. Корректные *workflow* определяют успешность промышленного предприятия, поэтому доказательство теоремы о корректности *workflow*, представленных в базах графических языков, позволит проектировщикам применять теорему и избежать дорогостоящих проектных ошибок на практике.

1 СМЕЖНЫЕ РАБОТЫ

Анализ современных работ [4–33] по тематике позволил сравнить полученные результаты с имеющимися, определить место и вклад настоящей работы в обработку потоков проектных работ. Ниже приведены выборочно статьи по анализу потоков проектных работ.

В работе [4] описана методология устранения ошибок в потоках проектных работ на основе определения шаблонов состояния, иерархии, детализации, однако способ выявления таких шаблонов не описан. Работа [5] исследует возможности проверки потоков проектных работ как атомарных, так и сложных событий в контексте временных автоматов (ТА) и временных сетей Петри (ТПН), сопоставляет семантику простых и сложных событий с событиями в ТА и ТПН. Простые события потоков проектных работ могут быть формализованы с помощью ТА и ТПН, но для сложных событий имеются ограничения в применении. Авторы работы [6] выполнили интерактивную визуализацию паттернов потоков работ с целью повышения наглядности в демонстрации заинтересованным сторонам (*Stakeholders*). В работе [7] использован известный метод *model checking* к логическим правилам, описывающим потоки проектных работ, а также предложены визуальные нотации к алгоритмам. Работа [8] демонстрирует применение *model checking* к проверке потоков проектных работ в разработке программного обеспечения, фокусируя внимание на электронной коммерции. Авторы работы [9] транслируют потоки проектных работ в YAWL-модели, анализируют их с помощью инструмента проверки WofYAWL, а также используют логистическую регрессию, чтобы показать, что сложность потоков проектных работ ока-

зывает значительное влияние на вероятность ошибки. В работе [10] предлагается упрощать потоки проектных работ и далее анализировать их с помощью сетей Петри, учитывая только свойство достижимости. Работа [11] содержит описание инструмента ProM 4.0 к анализу потоков проектных работ, базирующегося на *process mining*, т. е. извлечении информации о проектной деятельности по учтенным событиям, например, лог-файлу. Авторы работы [12] представляют потоки проектных работ без логического «И» в сети Петри и анализируют ее на ошибки, затратив на это полиномиальное время.

Таким образом, выявлено на основе источников, что отсутствуют теоремы к определению корректности потоков проектных работ в базах графических языков с линейной сложностью анализа, поэтому в настоящей работе представлена и доказана новая теорема.

2 ВВЕДЕНИЕ В ФОРМАЛИЗАЦИЮ

Определим на языке алгебры-логики роли, переменные и операции в хореографии потоков проектных работ в автоматизации проектирования сложных технических систем [1–3]. Положим Var_{Set} – множество содержит переменные k и l , причем $\tilde{k}$ обозначим кортеж переменных $\tilde{k} = \langle k_1, k_2, k_3, \dots, k_n \rangle$, где $k \in \tilde{k}$, а $\tilde{l}$ – кортеж переменных $\tilde{l} = \langle l_1, l_2, l_3, \dots, l_n \rangle$, где $l \in \tilde{l}$. Переменные могут принимать ограниченный набор значений, в том числе и двоичные значения. Положим $Operation_{Name}$ – множество содержит имена операций, $Operation_{Type} = \{one - way, request - response\}$ является множеством типов операций, где *one – way* обозначает одностороннюю операцию, тогда как *request – response* обозначает операцию запрос-ответ. Множество $Operation$ является набором операций, содержит кортеж из имени и типа операции: $Operation = \{ \langle name, type \rangle \mid name \in Operation_{Name}, Operation_{Type} \}$, где каждая операция однозначно идентифицируется по имени. Положим $Role_{Name}$ является множеством имен ролей, элементом которого является $\rho \in Role_{Name}$, а $Role$ является множеством ролей и определено как

$$Role = \{ (\rho, \omega, V) \mid \rho \in Role_{Name}, \omega \subseteq Operation, V \subseteq Var_{Set} \}.$$

Состояния системы переходов хореографии определены значениями переменных, вычисляемых по сле-

дующей формуле: $Function_{state} : Var_{Set} \rightarrow Value$, причем $Value \in D$. Положим $v \in Value \cup \emptyset$ и функция $Function_{state}(k)$ вычисляет значение переменной k в состоянии $state$, тогда $Function_{state}(k) = \emptyset$ означает, что k не задано, а запись $St[v/k]$ – система в состоянии St имеет переменную k со значением v . Аналогично запишем St – состояние в кортежном виде:

$$St[\tilde{v}/\tilde{k}]: St[v/k] = St',$$

$$St'(k') = \begin{cases} v, & \text{если } k' = k, \\ St(k') & \text{в любом другом случае.} \end{cases}$$

Форма Бэкуса-Наура – форма описания логических условий ограничений переменных позволит задать начальное состояние системы переходов в хореографии: $\chi ::= k \leq e \mid e \leq k \mid \neg \chi \mid \chi \wedge \chi$, где e обозначает выражение, содержащее переменные. Запись $e_{St} \rightarrow v$ обозначает, что выражение e вычислимо со значением v в состоянии St . Может быть использована запись $St \vdash \chi$ для обозначения того, что состояние St удовлетворяет условию χ . Правила для $\vdash$ определены следующим образом:

1. $St(k) = \emptyset \Rightarrow St \vdash (k \leq \emptyset \wedge \emptyset \leq k)$,
2. $e_{St} \rightarrow v, St(k) \leq v \Rightarrow St \vdash k \leq e$,
3. $e_{St} \rightarrow v, v \leq St(k) \Rightarrow St \vdash e \leq k$,
4. $St \vdash \chi' \wedge St \vdash \chi'' \Rightarrow St \vdash \chi' \wedge \chi''$,
5. $\neg(St \vdash \chi) \Rightarrow St \vdash \neg \chi$.

Правило (1) гласит, что в состоянии $St(k)$ выражение e определено со значением $\emptyset$, тогда и только тогда, когда $k = \emptyset$.

Поток проектных работ обмена C (*Conversation*) между ролями определен следующим образом:

$$C ::= 0 \mid \eta \mid C;$$

$$C \parallel C \parallel \left[\sum_{i \in N}^L (\eta_i \parallel C_i) \right] \left[\sum_{i \in N}^{\oplus L'} \chi_i?(\eta_i, C_i) \right],$$

$$\eta ::= (\rho_A, \rho_B, operation, \tilde{k}, \tilde{l}, dir) \mid k ::= e,$$

где η представляет поток проектных работ в обмене. Запись $::= (\rho_A, \rho_B, name, \tilde{k}, \tilde{l}, direction)$ означает, что обмен потоками проектных работ между ролями ρ_A, ρ_B прошел успешно, где $name \in Operation_{Name}$. Кортежи переменных $\tilde{k}, \tilde{l}$ используются для отсылки и приема в потоках проектных работ при обмене соответственно, $direction \in \{\uparrow, \downarrow\}$ определяет отправку ($\uparrow$) и прием ($\downarrow$) потоков проектных работ. $k ::= e$ означает, что вычисленное значение выражения e присвоено переменной k . η – потоки проектных работ в обмене могут принимать значение (0), имеет порядок последовательности ($C; C$), параллельности ($C \parallel C$), выбора в ситуации

принятия проектного решения: безусловный выбор (распараллеливание) $\sum_{i \in N}^L (\eta_i \parallel C_i)$, условный выбор (развилка) $\sum_{i \in N}^{\oplus L'} \chi_i?(\eta_i, C_i)$. Безусловный выбор выполняется независимо от условия χ . L является меткой слияния распараллеленных потоков проектных работ η_i и C_i , а L' является меткой слияния потоков проектных работ, заданных в развилке по условию χ .

Определим команду обмена как

$$\mu = \{action_C \mid action_C =$$

$$= (\rho_A, \rho_B, name, \tilde{v}, direction) \cup time_{clock}\},$$

где $(\rho_A, \rho_B, name, \tilde{v}, direction)$ представляет характеристики потока проектных работ, $time_{clock}$ – темпоральный параметр. Запись $(C, St) \xrightarrow{\mu} (C', St')$ означает, что произошла команда μ , результатом которой стал один поток проектных работ C' , и система хореографии из состояния St перешла в St' .

3 ОПРЕДЕЛЕНИЕ ГРАФ-СХЕМЫ ПОТОКОВ ПРОЕКТНЫХ РАБОТ

АМ-модель [1] представим для наглядности в виде $G(AM)$ граф-схемы потоков проектных работ в САПР: $G(AM) = (S, WF)$, где $S = \{s_i \mid i = \overline{1, N}\}$ – множество состояний; $WF = \{wf_i \mid i = \overline{1, N}\}$ – множество потоков проектных работ, $WF \subseteq S \times S$; $\sum^{\parallel}$ – подмножество состояний распараллеливания, $\sum^{\parallel} \subseteq S$; L – подмножество состояний слияния распараллеленных потоков проектных работ, $L \subseteq S$; $\sum^{\oplus}$ – подмножество состояний развилки, $\sum^{\oplus} \subseteq S$; L' – подмножество состояний слияния потоков проектных работ развилки, $L' \subseteq S$.

Каждому wf_i ставится один из типов потоков проектных работ в соответствие с таблицей 1. «Поток проектных работ № 1 – Отправка» содержит (ρ_A, ρ_B) – две роли, $\mu_{\uparrow \tilde{\delta}}$ – команду отправки, $\eta_{\uparrow \tilde{\delta}}$ – поток проектных работ отправки, $S[\tilde{v}/\tilde{k}]$ – начальное состояние и $S'[\tilde{\delta}/\tilde{l}]$ – конечное состояние. «Поток проектных работ № 2 – Прием» содержит (ρ_A, ρ_B) – две роли, $\mu_{\downarrow \tilde{v}}$ – команду приема, $\eta_{\downarrow \tilde{v}}$ – поток проектных работ приема, $S[\tilde{\delta}/\tilde{l}]$ – начальное состояние и $S'[\tilde{v}/\tilde{k}]$ – конечное состояние. «Поток проектных работ № 3 – Назначение» содержит команду τ для назначения переменной l величины v , причем $S[v/l]$ конечным состоянием будет. «Поток проектных работ № 4 – Последовательность» содержит команду μ и I, I', D – три последовательных потока работ соответственно. «Поток проектных работ № 5 – Параллельность» содержит команду μ и I, I' –

пару последовательных потоков проектных работ, а также D – параллельный им поток. «Поток проектных работ № 6 – Выбор 1 (распараллеливание)» содержит I_i , θ_i – исходящие потоки проектных работ в состоянии S , участвующих в конъюнкции $\sum_{i \in N} \theta_i \parallel I_i$, команду μ , состояние S' с θ'_i – исходящим потоком проектных работ, состояние S'' с I'_i – исходящим потоком проектных работ.

работ. «Поток проектных работ № 7 – Выбор 2 (развилка)» содержит команду μ ; θ_i , I_i – потоки проектных работ в состоянии S , участвующих в ϕ_i – строгом дизъюнктивном условии по выбору дальнейшего состояния; состояние S' с θ'_i – исходящим потоком проектных работ, состояние S'' с I'_j – исходящим потоком проектных работ.

Таблица 1

Соответствие типа потока проектных работ $G(AM)$

№	Тип потока проектных работ	$G(AM)$
1	Поток проектных работ № 1 – Отправка	
2	Поток проектных работ № 2 – Прием	
3	Поток проектных работ № 3 – Назначение	
4	Поток проектных работ № 4 – Последовательность	
5	Поток проектных работ № 5 – Параллельность	
6	Поток проектных работ № 6 – Выбор 1 (распараллеливание)	
7	Поток проектных работ № 7 – Выбор 2 (развилка)	

4 КОРРЕКТНОСТЬ $G(AM)$

Определение 1. $G(AM)$ называется корректной, если определяемый ею wf_G поток работ удовлетворяет следующим условиям:

1. wf_G детерминирован, т. е. если wf'_G и wf''_G – различные реализации потоков проектных работ, то $\forall d_k \in D(wf'_G(d_k) = wf''_G(d_k))$ – «Свойство детерминированности предполагает, что алгоритм описывается на таком формальном языке, представлен в виде такой четко сформулированной системы правил, что результаты его работы не зависят от того, какая именно система будет осуществлять эти правила» [13].

2. Продолжительность потока проектных работ при применении его к любым исходным данным конечна: $\forall d_k \in D(\tau(wf'_G(d_k)) < \infty)$.

К нарушению корректности $G(AM)$ приводят следующие шаблонные конфигурации:

- дедлок (deadlock) или «тупик» в состоянии L (рис. 1а), в $G(AM)$ существует пара состояний S_i, S_j , таких, что образуют контур $S_i \rightarrow wf_i S_j, S_j \rightarrow wf_j S_p, S_p \rightarrow wf_p L, S_p \rightarrow wf_{p'}, L \rightarrow wf_k S_i, S_i \rightarrow wf_a L, wf_i \parallel wf_{p'}$, причем потоки проектных работ $wf_i, wf_j, wf_p, wf_{p'}, wf_k, wf_a$ не содержат состояния $L \rightarrow \sum_{i \in N}^{\oplus L'} wf_i \oplus wf_a? wf_k S_i$, где $\oplus$ – строгая дизъюнкция;

- неоднозначность в $G(AM)$ в состоянии L' (рис. 1б), существует пара таких состояний S_i, S_j , что $[wf_i S_i] \parallel [wf_j S_j]$, и состояние L' такое, что $S_i \rightarrow \sum_{i \in N}^{\oplus L'} wf_i \oplus wf_j? wf_k L', S_j \rightarrow \sum_{i \in N}^{\oplus L'} wf_i \oplus wf_j? wf_k L'$;

- зависание в $G(AM)$ в состоянии L (рис. 1в), существует пара таких состояний S_i, S_j , что $[wf_i S_i] \oplus [wf_j S_j]$, и состояние L такое, что $S_i \rightarrow \sum_{i \in N}^{\parallel L} wf_i \parallel wf_j L, S_j \rightarrow \sum_{i \in N}^{\parallel L} wf_i \parallel wf_j L$.

Бывают ситуации, в которых между двумя состояниями S_1, S_2 не существует причинно-следственных потоков проектных работ wf_1, wf_2 и $S_1 \parallel S_2$. При этом недопустимо, чтобы $wf_1 S_1$ пересекалось во времени с $wf_2 S_2$. Эти ситуации обычно вызваны необходимостью использовать один и тот же ресурс и называются *ресурсными конфликтами*. Они могут быть устранены введением приоритета и управляющего механизма – арбитра. Установление приоритета для wf_1 и wf_2 , заключается в том, что выход S_1 соединяется со входом S_2 путем включения двух состояний распараллеливания $\sum^{\parallel}$ и слияния параллельных потоков проектных работ L (рис. 2а). Введение арбитра заключается в добавле-

Рис. 1. Экземпляры $G(AM)$: «тупик» (а), неоднозначность (б), зависание (в)

ние нового специального состояния, которое должно соответствовать появлению в реальной ситуации арбитра [14, 15] и первым дать выполнение того потока проектных работ, для которого исходные данные оказались готовы раньше, и запретить выполнять второй поток проектных работ, пока не завершится выполняться первый (рис. 2б).

Основными трудностями при разработке потоков проектных работ является построение корректных

$G(AM)$. Приемы, представленные в работе [16], могут быть применимы к построению корректных $G(AM)$, причем выполнение указанных приемов гарантирует получение хорошо сформулированных $G(AM)$ (well formed) или WF -сетей, которые определяются следующим образом: а) всякая $G(AM)$ без циклов и контуров представляет собой WF -сеть; б) представленные ниже три $G(AM)$ являются WF -сетями (рис. 3); в) $G(AM)$,

Рис. 2. В $G(AM)$ для $wf_p, wf_{p'}$ установление приоритета (а), введение арбитра A (б)

Рис. 3. Типовые $G(AM)$, являющиеся WF -сетями

полученная заменой элементов на WF -сети, является WF -сетью.

Построение $G(AM)$ с учетом обозначенных приемов обеспечивает корректность, структурированность и облегчает последующий синтез исходного кода. Однако класс WF -сети является ограниченным классом корректных $G(AM)$, существует много типовых корректных проектных конфигураций $G(AM)$ (рис. 4), но не принадлежащих к классу WF -сети. Например, невозможно остаться в рамках WF -сети, если необходимо установить приоритеты, поэтому задача проверки $G(AM)$ ставится шире, чем в работе [16], и решается на основе установления соответствия между корректностью $G(AM)$ и свойством живости, безопасности ее сети Петри.

Чтобы установить это соответствие необходимо преобразовать $G(AM)$ в граф-схемы алгоритмов (ГСА) и воспользоваться доказанной теоремой № 2.1 работы [17] о том, что ГСА является корректной тогда и только тогда, когда ее сеть Петри имеет свойства безопасности и живости. $G(AM)$ преобразуем в ГСА с помощью формул перехода [18], имеющих в общем виде следующую запись: $S \rightarrow \alpha S'$, где S – начальное состояние системы; S' – конечное состояние системы; α – условие в виде логического выражения. Формулы перехода для $G(AM)$ имеют следующий вид:

- для последовательного выполнения потоков проектных работ $I' : S \rightarrow I'S'$, где S – исходное состояние; S' – целевое состояние; I' – поток проектных работ;
- для выполнения потоков проектных работ № 7 –

Выбор 1 (распараллеливание): $S \rightarrow \sum_{i \in N}^{\parallel L} \theta'_i S'_i$,

Рис. 4. Типовая корректная проектная конфигурация $G(AM)$

где S – исходное состояние; S' – целевые состояния; $\sum_{i \in N}^{\parallel L}$ – распараллеливание потоков проектных работ; L – место (состояние) слияния настоящих потоков проектных работ; θ'_i – параллельные потоки проектных работ;

- для слияния параллельных потоков проектных работ θ'_i в состоянии $L : S_i \rightarrow \theta'_i L$, где S_i – исходные состояния; S – целевое состояние; L – место (состояние) слияния настоящих потоков проектных работ; θ'_i – параллельные потоки проектных работ; $i \in \overline{1.N}$;

- для выполнения потоков проектных работ № 8 – Выбор 2 (развилка): $S \rightarrow \sum_{i \in N}^{\oplus L'} \phi_i ? \theta_i S'_i$, где S – исходное состояние; S'_i – целевые состояния; L' – место (состояние) слияния настоящих потоков проектных работ; ϕ_i – логическое выражение условия развилки; θ_i – потоки проектных работ развилки;

- для слияния потоки проектных работ развилки θ_i в состоянии $L' : S_i \rightarrow \theta'_i L'$, где S_i – исходное состояние; S – целевое состояние; L' – место (состояние) слияния настоящих потоков проектных работ; $i \in \overline{1.N}$.

Формулы перехода для ГСА имеют следующий вид:

- для последовательного выполнения операторов $A_i : A_i \rightarrow A_{i+1}$, где $i \in \overline{1.N}$;

- для параллельного выполнения операторов $A_j : A_i \rightarrow \wedge^L A_j$, где $\wedge$ – логическое «И», L – место (оператор) слияния веток алгоритма; $i, j \in \overline{1.N}$;

- для слияния параллельных веток алгоритма: $A_i \rightarrow L A_j$, где L – место (оператор) слияния параллельных веток алгоритма; $i, j \in \overline{1.N}$;

- для выполнения развилки в алгоритме: $A_i \rightarrow \vee^{L'} A_j$, где $\vee$ – логическое «ИЛИ», L' – место (оператор) слияния веток развилки алгоритма; $i, j \in \overline{1.N}$;

- для слияния веток развилки в алгоритме: $A_i \rightarrow L' A_j$, где L' – место (оператор) слияния веток развилки в алгоритме; $i, j \in \overline{1.N}$.

5 ФОРМУЛИРОВКА И ДОКАЗАТЕЛЬСТВО ТЕОРЕМЫ КОНСТРУКТИВНЫМ ПОДХОДОМ

Теорема. $G(AM)$ можно преобразовать в ГСА. ГСА можно преобразовать в $G(AM)$.

Доказательство теорем проведем, применяя конструктивный подход, а именно:

- 1) путем построения алгоритмов преобразования соответственно $G(AM)$ в ГСА, ГСА в $G(AM)$ с установкой меток (*Mark*) к состояниям (вершинам), потокам проектных работ (связям);

- 2) выполнением следующих условий при расстановке и учете меток:

а) $G(AM)$ в ГСА

$$\sum_{i=1}^n \text{Mark}[S_i] + \sum_{j=1}^m \text{Mark}[wf_j] = 0,$$

где $n \in \overline{1.N}$ является числом состояний, $m \in \overline{1.N}$ является числом потоков проектных работ. Сумма всех меток $\text{Mark}[S_i]$ и $\text{Mark}[wf_j]$ после преобразования равна нулю.

б) ГСА в $G(AM)$

$$\sum_{i=1}^n \text{Mark}[V_i] + \sum_{j=1}^m \text{Mark}[E_j] = 0,$$

где $n \in \overline{1.N}$ является числом вершин, $m \in \overline{1.N}$ является числом связей. Сумма всех меток $\text{Mark}[V_i]$ и $\text{Mark}[E_j]$ после преобразования равна нулю.

Теоретико-множественная модель ГСА имеет следующий вид: $G = (V, E, TV, TE)$, где $V = \{v_i | i \in \overline{1.N}\}$ – множество вершин; $E = \{e_i | i \in \overline{1.N}\}$ – множество связей, $E \subseteq V \times V$; $TV = \{tv_i | i \in \overline{1.N}\}$ – множество типов вершин; $TE = \{te_i | i \in \overline{1.N}\}$ – множество типов связей. Введем ограничение на входные данные: ГСА может содержать только один тип направленной связи, которая соответствует потоку проектных работ в $G(AM)$.

Алгоритм преобразования G в $G(AM)$

1. Определение формул перехода для вершин G в $G(AM)$

В G необходимо определить формулы перехода для каждого типа вершины и связи в $G(AM)$: $Ftrans : V \cup E \rightarrow G(AM)$. Для каждой входящей связи e должен существовать $wf_in[N]$ – входной поток проектных работ в $G(AM)$, и для каждой исходящей связи должен существовать $wf_out[N]$ – исходящий поток проектных работ.

2. Формирование множеств $AVN[N]$

Для каждой вершины $v_i \in V$, для каждой связи $e_i \in E$ в G формируем преобразование в $G(AM)$, получаем множества $VN : s_i \in VN$, $AVN[N] : AVN = \{Ftrans(item_i) | item_i \in V \cup E\}$, где $Ftrans$ – формула перехода с расстановкой меток, имеющая следующие процедуры:

1) процедура анализа G на базе RVTI-анализатора с линейной сложностью анализа [34] и расстановки $Mark_i \in \overline{1.N}$ меток в каждой пройденной $v_i \in V$ и $e_i \in E$;

2) процедура анализа G на базе RVTI-анализатора с линейной сложностью и удаления меток в каждой пройденной $v_i \in V$ и $e_i \in E$ при выполнении следующего условия – существует соответствие к $G(AM)$?

Сумма меток после выполнения $Ftrans$, отличная от нуля, означает ошибку в преобразовании G в $G(AM)$.

3. Объединение множеств $AVN[N]$ в $G(AM)$

Положим $G(AM) = (\emptyset, \emptyset)$. Скопируем все потоки проектных работ, все состояния в $G(AM)$:

$$G(AM)[S] = \{s | \exists a \in AVN, s \in a[N][V]\},$$

$$G(AM)[wf] = \{wf | \exists a \in AVN, wf \in a[N][E]\},$$

$$G(AM)[\sum^{\parallel}] = \{s | \exists a \in AVN, s \in a[N][V]\},$$

$$G(AM)[L] = \{s | \exists a \in AVN, s \in a[N][V]\},$$

$$G(AM)[\sum^{\oplus}] = \{s | \exists a \in AVN, s \in a[N][V]\},$$

$$G(AM)[L'] = \{s | \exists a \in AVN, s \in a[N][V]\}.$$

4. Преобразование потоков проектных работ в $G(AM)$

На основе каждой связи $e_i \in E$ соединяем состояния в $G(AM)$. Для e_i находим начальную v_1 и конечную v_2 вершины, и соответствующие им состояния $s_1 \in VN$, $s_2 \in VN$. Соединяем потоками проектных работ выбранные состояния: $\exists wf_in[N] \in AVN$, $\exists wf_out[N] \in AVN$, $\exists wf_in[N][s_2] = e_i$, $\exists wf_out[N][s_1] = e_i$.

5. Конец

На выходе преобразования получаем $G(AM)$. Вершины G преобразованы в состояния $G(AM)$, связи G преобразованы в соответствующие потоки проектных работ в $G(AM)$.

Алгоритм преобразования $G(AM)$ в G

1. Определение формул перехода $G(AM)$ в G

Необходимо определить формулу перехода для каждого типа вершины и связи $G(AM)$ в G : $Ftrans : S \cup WF \rightarrow G$. Для каждого потока проектных работ $wf_in[N]$ должна существовать $e_in[N]$ – входная связь в G , и для каждого потока проектных работ $wf_out[N]$ должна существовать $e_out[N]$ – исходящая связь.

2. Формирование множеств AVN

Для каждой вершины $v_i \in V$, связи $e_i \in E$ в G формируем преобразование из $G(AM)$ и получаем множества: $VN : s_i \in VN$, $AVN[N] : AVN = \{Ftrans(item_i) | item_i \in S \cup WF\}$, где $Ftrans$ – формула перехода с расстановкой меток, имеющая следующие процедуры:

1) процедура анализа $G(AM)$ на базе RVTI-анализатора с линейной сложностью анализа [34] и расстановки $Mark_i \in \overline{1.N}$ меток в каждой пройденной $s_i \in S$ и $wf_i \in WF$;

2) процедура анализа $G(AM)$ на базе RVTI-анализатора с линейной сложностью анализа и удаления меток в

каждой пройденной $s_i \in S$ и $wf_i \in WF$ при выполнении следующего условия – существует соответствие к G ?

Сумма меток после выполнения $Ftrans$, отличная от нуля, означает ошибку в преобразовании $G(AM)$ в G .

3. Объединение множеств $AVN[N]$ в G

Положим $G = (\emptyset, \emptyset, \emptyset, \emptyset)$. Скопируем все e_i связи и v_i вершины в G :

$$G[V] = \{v_i | \exists a \in AVN, v_i \in a[N][V]\},$$

$$G[E] = \{e_i | \exists a \in AVN, e_i \in a[N][E]\},$$

$$G[TE] = \{te_i | \exists a \in AVN, te_i \in a[N][TE]\},$$

$$G[TV] = \{tv_i | \exists a \in AVN, tv_i \in a[N][TV]\}.$$

4. Преобразование связей в G

На основе каждого потока проектных работ wf_n соединяем вершины в G . Для каждого потока проектных работ wf_n находим начальное $s_1 \in VN$ и конечное $s_2 \in VN$ состояния и соответствующие им вершины в G : $v_1 \in V$, $v_2 \in V$. Ставим связи к выбранным вершинам v_1 , v_2 : $\exists e_in[N] \in AVN$, $\exists e_out[N] \in AVN$, $e_in[N][v_2] = wf_n e_out[N][v_1] = wf_n$.

5. Конец ■

ЗАКЛЮЧЕНИЕ

Теорема о корректности $G(AM)$ доказана на основе конструктивного подхода. Корректные потоки проектных работ, в том числе алгоритмы, позволяют значительно уменьшить число ошибок при разработке сложных технических систем на этапе эскизного макетирования, обеспечат экономический эффект промышленному предприятию.

СПИСОК ЛИТЕРАТУРЫ

1. Войт Н.Н. Принцип «ансамбль» и аналитическая модель организации потоков проектных работ // Автоматизация процессов управления. 2020. № 3 (61). С. 124–137.
2. Войт Н.Н. Экземпляр аналитической модели организации потоков проектных работ в условиях цифрового производства // Автоматизация процессов управления. 2020. № 4 (62). С. 118–130.
3. Разработка и исследование информационных потоков в системах проектирования и согласования конструкторской документации при проектировании сложных технических изделий / Н.Н. Войт, М.Е. Уханова, Д.С. Канев, А.С. Степанов, В.А. Гордеев // Информационно-измерительные и управляющие системы. 2020. Т. 18, № 1. С. 6–16.
4. Repa V., Bruckner T. Methodology for modeling and analysis of business processes (MMABP) // J. Syst. Integr. 2015. Vol. 6, no. 4. pp. 17–28.
5. Verification of event-driven process chain with timed automata and time Petri nets / A. Amjad, F. Azam,

M.W. Anwar and W.H. Buttm // Proc. 9th IEEE GCC Conf. Exhib., May 2017.

6. Providing EA decision support for stakeholders by automated analyses / D. Jugel, S. Kehrer, C.M. Schweda and A. Zimmermann // Digital Enterprise Computing. Bonn, Germany : Gesellschaft für Informatik, 2015.

7. Feja S., Fötsch D. Model checking with graphical validation rules // Proc. 15th Annu. IEEE Int. Conf. Workshop Eng. Comput. Based Syst. (ECBS). Mar./Apr. 2008. pp. 117–125.

8. Pfeiffer J.-H., Rossak W.R., Speck A. Applying model checking to workflow verification // Proc. 11th IEEE Int. Conf. Workshop Eng. Comput.-Based Syst. (ECBS). May 2004. pp. 144–151.

9. Detection and prediction of errors in EPCs of the SAP reference model / J. Mendling, H.M.W. Verbeek, B.F. van Dongen, W.M.P. van der Aalst and G. Neumann // Data Knowl. Eng. 2008. Vol. 64, no. 1. pp. 312–329.

10. Van Dongen B.F., van der Aalst W.M.P., Verbeek H.M.W. Verification of EPCs: Using reduction rules and Petri nets // Proc. 17th Int. Conf. Adv. Inf. Syst. Eng. (CAISE). 2005. pp. 327–386.

11. ProM 4.0: Comprehensive support for real process analysis / W.M.P. van der Aalst et al. // Proc. 28th Int. Conf. Appl. Theory Petri Nets Other Models Concurrency (ICATPN). 2007. pp. 484–494.

12. Van der Aalst W.M.P. Formalization and verification of event-driven process chains // Inf. Softw. Technol. 1999. Vol. 41, no. 10. pp. 639–650.

13. Напалков А.В. Мозг человека и искусственный интеллект. URL: <https://www.ngpedia.ru/pg4851178uaHVxYp0001412943> (дата обращения: 31.05.2021).

14. Plummer W.W. Asynchronous artiters // IEEE Transactions of Computers. 1972. V. C-21, no 1. pp. 37–42.

15. Corsini P., Frosini G. A model for Asynchronous Control Networks // Digital Processes. 1976. N 2. pp. 47–62.

16. Rumbaugh J. A Data Flow Multiprocessor // IEEE Transaction on Computers. 1977. V. C-26, no 1. pp. 138–147.

17. Методы параллельного микропрограммирования / П.А. Анишев, С.М. Ачасова, О.Л. Бандман [и др.]. Новосибирск : Наука, 1981. 184 с

18. Баранов С.И. Синтез микропрограммных автоматов. Л., 1979.

19. Виролайнен А.В. Моделирование бизнес-процессов // Лучшая научная статья 2019 : сб. ст. XXIV междунар. науч.-исслед. конкурса. Пенза, 2019. С. 82–86.

20. Чернышов А.С. Анализ методов моделирования автоматизированных систем // Информационные технологии и автоматизация управления : сб. тр. конф. (Омск, 15–16 мая 2019 г.). Омск, 2019. С. 363–370.

21. Копп А.М., Орловский Д.Л. Подход к анализу и оптимизации моделей бизнес-процессов в нотации BPMN // Радіоелектроніка, інформатика, управління. 2018. № 2 (45). С. 108–116.

22. Сергиевский М.В., Кирпичникова К.К. Валидация и оптимизация диаграмм классов UML // *Cloud of science*. 2018. Т. 5, № 2. С. 367–378.
23. Богдан И., Задорожний А. Классификация ошибок на UML-диаграммах, возникающих в ходе разработки ИТ-проектов // *Технічні науки та технології*. 2018. № 1. С. 68–78.
24. Копп А.М., Орловский Д.Л. Анализ та оптимізація моделей бізнес-процесів у нотаціях EPC та BPMN // *Технічні науки та технології*. 2018. №. 4 (14). С. 145–152.
25. Design Time Validation for the Correct Execution of BPMN Collaborations / J. Anseeuw et al. // *CLOSER* (1). 2016. pp. 49–58.
26. A Pattern Based Method for Simplifying a BPMN Process Model / M. Ramos-Merino et al. // *Applied Sciences*. 2019. Vol. 9, № 11. P. 2322.
27. Claes J., Vandecaveye G. The impact of confusion on syntax errors in simple sequence flow models in BPMN // *International Conference on Advanced Information Systems Engineering*. Springer, Cham, 2019. pp. 5–16.
28. Lima L., Tavares A., Nogueira S.C. A framework for verifying deadlock and nondeterminism in UML activity diagrams based on CSP // *Science of Computer Programming*. 2020. P. 102497.
29. Huang H., Peng R., Feng Z. Efficient and exact query of large process model repositories in cloud workflow systems // *IEEE Transactions on Services Computing*. 2015. Vol. 11, № 5. pp. 821–832.
30. van der Aalst W.M.P. Everything You Always Wanted to Know About Petri Nets, but Were Afraid to Ask // *International Conference on Business Process Management*. Springer, Cham, 2019. pp. 3–9.
31. Barash M., Okhotin A. Generalized LR parsing algorithm for grammars with one-sided contexts // *Theory of Computing Systems*. 2017. Vol. 61, № 2. pp. 581–605.
32. Formal languages over GF (2) / E. Bakinova et al. // *International Conference on Language and Automata Theory and Applications*. Springer, Cham, 2018. pp. 68–79.
33. Gorokhov A., Grigorev S. Extended Context-Free Grammars Parsing with Generalized LL // *International Conference on Tools and Methods for Program Analysis*. Springer, Cham, 2017. pp. 24–37.
34. Afanasyev A.N., Voit N.N., Kirillov S.Y. Temporal automata RVTI-grammar for processing diagrams in visual languages as BPMN, EEPN and ASKON-Volga // *ACM International Conference Proceeding Series* : 5, Istanbul, 16–17 apr. 2019. Istanbul, 2019. pp. 71–75. DOI 10.1145/3323933.3324067.
2. Voit N.N. Ekzempliar analiticheskoi modeli organizatsii potokov proektnykh rabot v usloviakh tsifrovogo proizvodstva [The Instance of Analytical Model for Organization in Design Workflows]. *Avtomatizatsiia protsessov upravleniia* [Automation of Control Processes], 2020, no. 4 (62), pp. 118–130.
3. Voit N.N., Ukhanova M.E., Kanev D.S., Stepanov A.S., Gordeev V.A. Razrabotka i issledovanie informatsionnykh potokov v sistemakh proektirovaniia i soglasovaniia konstruktorskoi dokumentatsii pri proektirovanii slozhnykh tekhnicheskikh izdelii [Development and Research of Information Flows in Design Systems and Agreement of Design Documents for Designing Complex Technical Products]. *Informatsionno-izmeritelnye i upravliaiushchie sistemy* [Information-Measuring and Control Systems], 2020, vol. 18, no. 1, pp. 6–16.
4. Repa V., Bruckner T. Methodology for Modeling and Analysis of Business Processes (MMABP). *J. Syst. Integr.*, 2015, vol. 6, no. 4, pp. 17–28.
5. Amjad A., Azam F., Anwar M.W., Buttm W.H. Verification of Event-Driven Process Chain with Timed Automata and Time Petri Nets. *Proc. 9th IEEE GCC Conf. Exhib.* May 2017.
6. Jugel D., Kehrer S., Schweda C.M., Zimmermann A. Providing EA Decision Support for Stakeholders by Automated Analyses. *Digital Enterprise Computing*. Bonn, Germany, Gesellschaft für Informatik Publ., 2015.
7. Feja S., Fötsch D. Model Checking with Graphical Validation Rules. *Proc. 15th Annu. IEEE Int. Conf. Workshop Eng. Comput. Based Syst. (ECBS)*. Mar./Apr. 2008, pp. 117–125.
8. Pfeiffer J.-H., Rossak W.R., Speck A. Applying Model Checking to Workflow Verification. *Proc. 11th IEEE Int. Conf. Workshop Eng. Comput.-Based Syst. (ECBS)*. May 2004, pp. 144–151.
9. Mendling J., Verbeek H.M.W., B.F. van Dongen, W.M.P. van der Aalst, G. Neumann. Detection and Prediction of Errors in EPCs of the SAP Reference Model. *Data Knowl. Eng.*, 2008, vol. 64, no. 1, pp. 312–329.
10. Van Dongen B.F., van der Aalst W.M.P., Verbeek H.M.W. Verification of EPCs: Using Reduction Rules and Petri Nets. *Proc. 17th Int. Conf. Adv. Inf. Syst. Eng. (CAiSE)*. 2005, pp. 327–386.
11. W.M.P. van der Aalst et al. ProM 4.0: Comprehensive Support for Real Process Analysis. *Proc. 28th Int. Conf. Appl. Theory Petri Nets Other Models Concurrency (ICATPN)*. 2007, pp. 484–494.
12. Van der Aalst W.M.P. Formalization and Verification of Event-Driven Process Chains. *Inf. Softw. Technol.*, 1999, vol. 41, no. 10, pp. 639–650.
13. Napalkov A.V. *Mozg cheloveka i iskusstvennyi intellekt* [Human Computer and Artificial Intelligence]. Available at: <https://www.ngpedia.ru/pg4851178uaHVxYp0001412943> (accessed 31.05.2021).
14. Plummer W.W. Asynchronous Ertiters. *IEEE Transactions of Computers*, 1972, vol. C-21, no 1, pp. 37–42.

REFERENCES

1. Voit N.N. Printsip "Ansambli" i analiticheskai model organizatsii potokov proektnykh rabot [The Ensemble Principle and Analytical Model of Project Workflow Organization]. *Avtomatizatsiia protsessov upravleniia* [Automation of Control Processes], 2020, no. 3 (61), pp. 124–137.

15. Corsini P., Frosini G. A Model for Asynchronous Control Networks. *Digital Processes*, 1976, no. 2, pp. 47–62.
16. Rumbaugh J. A Data Flow Multiprocessor. *IEEE Transaction on Computers*, 1977, vol. C-26, no. 1, pp. 138–147.
17. Anishev P.A., Achasova S.M., Bandman O.L. et al. *Metody parallelnogo mikroprogramirovaniia* [Microcoding Techniques]. Novosibirsk, Nauka Publ., 1981. 184 p.
18. Baranov S.I. *Sintez mikroprogrammnykh avtomatov* [Synthesis of Microprogram Automation]. Leningrad, 1979.
19. Virolainen A.V. Modelirovanie biznes-protsessov [Business Process Modeling]. *Luchshaia nauchnaia statia 2019. Sb. st. XXIV mezhdunar. nauch.-issled. konkursa* [Proc. of the 24th Int. Sci. and Research Comp. The Best Research Paper in 2019]. Penza, Nauka i Prosveshchenie Publ., 2019, pp. 82–86.
20. Chernyshov A.S. Analiz metodov modelirovaniia avtomatizirovannykh sistem [Analysis of Modeling Computer-Aided Systems]. *Informatsionnye tekhnologii i avtomatizatsiia upravleniia. Sb. tr. konf.* (Omsk, 15–16 may, 2019) [Proc. of Conf.]. Omsk, 2019, pp. 363–370.
21. Kopp A.M., Orlovskiy D.L. Podkhod k analizu i optimizatsii modelei biznes-protsessov v notatsii BPMN [Design Automation of Easy-Tested Digital Finite State Machines]. *Radioelektronika, informatika, upravlinnia* [Radio Electronics, Computer Science, Control], 2018, no. 2 (45), pp. 108–116.
22. Sergievskii M.V., Kirpichnikova K.K. Validatsiia i optimizatsiia diagramm klassov UML [Validating and Optimizing UML Class Diagrams]. *Cloud of Science*, 2018, vol. 5, no. 2, pp. 367–378.
23. Bogdan I., Zadorozhnyi A. Klassifikatsiia oshibok na UML-diagrammakh, voznikaushchikh v khode razrabotki IT-proektov [The Classification of Errors on UML-Diagrams Occurring in the Development of IT-Projects]. *Tekhnichni nauki ta tekhnologii* [Technical Sciences and Technologies], 2018, no. 1, pp. 68–78.
24. Kopp A.M., Orlovskii D.L. Analiz ta optimizatsiia modelei biznes-protsesiv u notatsiiakh EPC ta BPMN [Analysis and Optimization of EPC and BPMN Business Process Models]. *Tekhnichni nauki ta tekhnologii* [Technical Sciences and Technologies], 2018, no. 4 (14), pp. 145–152.
25. Anseeuw J. et al. Design Time Validation for the Correct Execution of BPMN Collaborations. *CLOSER* (1), 2016, pp. 49–58.
26. Ramos-Merino M. et al. A Pattern Based Method for Simplifying a BPMN Process Model. *Applied Sciences*, 2019, vol. 9, no. 11, p. 2322.
27. Claes J., Vandecaveye G. The Impact of Confusion on Syntax Errors in Simple Sequence Flow Models in BPMN. *International Conference on Advanced Information Systems Engineering*. Springer, Cham, 2019, pp. 5–16.
28. Lima L., Tavares A., Nogueira S.C. A Framework for Verifying Deadlock and Nondeterminism in UML Activity Diagrams Based on CSP. *Science of Computer Programming*, 2020, p. 102497.
29. Huang H., Peng R., Feng Z. Efficient and Exact Query of Large Process Model Repositories in Cloud Workflow Systems. *IEEE Transactions on Services Computing*, 2015, vol. 11, no. 5, pp. 821–832.
30. Van der Aalst W.M.P. Everything You Always Wanted to Know About Petri Nets, but Were Afraid to Ask. *International Conference on Business Process Management*. Springer, Cham, 2019, pp. 3–9.
31. Barash M., Okhotin A. Generalized LR Parsing Algorithm for Grammars with One-Sided Contexts. *Theory of Computing Systems*, 2017, vol. 61, no. 2, pp. 581–605.
32. Bakinova E. et al. Formal Languages over GF (2). *International Conference on Language and Automata Theory and Applications*. Springer, Cham, 2018, pp. 68–79.
33. Gorokhov A., Grigorev S. Extended Context-Free Grammars Parsing with Generalized LL. *International Conference on Tools and Methods for Program Analysis*. Springer, Cham, 2017, pp. 24–37.
34. Afanasev A.N., Voit N.N., Kirillov S.Y. Temporal Automata RVTI-Grammar for Processing Diagrams in Visual Languages as BPMN, EEPK and ASKON-Volga. *ACM International Conference Proceeding Series 5*. Istanbul, 16–17 apr. 2019. Istanbul, 2019, pp. 71–75. DOI 10.1145/3323933.3324067.