МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«УЛЬЯНОВСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Н. Н. ВОЙТ, А. Н. АФАНАСЬЕВ

Структурно-семантический анализ потоков работ и обучение проектировщиков в автоматизации проектирования

Ульяновск
УлГТУ
2018

УДК 658.512 : 004.896
ББК 30.2-5-05
В 65

Научный редактор д-р техн. наук, профессор Афанасьев А. Н.
Рецензенты:
начальник управления информационных технологий АО «Ульяновский механический завод» Войт А. Н.;
канд. техн. наук, начальник Центра подготовки, переподготовки персонала и специалистов инозаказчика АО «Ульяновский механический завод» Гульшин В. А.

Войт, Николай Николаевич
В 65 Структурно-семантический анализ потоков работ и обучение проектировщиков в автоматизации проектирования /
Н. Н. Войт, А. Н. Афанасьев. – Ульяновск : УлГТУ, 2018. – 142 с.
ISBN 978-5-9795-1872-5

В монографии рассмотрены вопросы анализа диаграмматических моделей потоков работ в автоматизации проектирования сложных технических систем, а также преобразования этих потоков работ с целью повышения успешности разработки автоматизированных систем. Определены структурные и семантические ошибки диаграмматических моделей потоков работ. Базовым математическим аппратом формального анализа потоков работ взята темпоральная RVTI-грамматика. Изложены методы синтаксического и семантического анализа, нейтрализации ошибок, трансляции и метатрансляции. Работа содержит материал по моделям и методу адаптивного обучения проектировщиков структурно-семантическому анализу и преобразованию потоков работ в автоматизации проектирования в условиях крупного радиотехнического предприятия, представлен успешный проект обучения.

Монография может быть полезна студентам старших курсов, магистрантам, аспирантам и специалистам в области разработки автоматизированных систем, подготовлена на кафедре «Вычислительная техника» УлГТУ.

Исследования поддержаны грантом Министерства образования и науки Российской Федерации, проект №2.1615.2017/4.6. Исследование выполнено при финансовой поддержке РФФИ и Правительства Ульяновской области в рамках научного проекта №16-47-732152. Исследование выполнено при финансовой поддержке РФФИ в рамках научного проекта №17-07-01417. Исследование выполнено при финансовой поддержке РФФИ и Ульяновской области в рамках научного проекта №18-47-730032.

УДК 658.512 : 004.896
ББК 30.2-5-05

ISBN 978-5-9795-1872-5

ОГЛАВЛЕНИЕ

Введение

Работа содержит методы, модели, алгоритмы и средства двух предметных областей – автоматизации проектирования сложных технических систем в условиях крупной проектной организации и автоматизации процесса обучения проектировщиков без отрыва от производства. Представим анализ современного состояния исследований в каждой предметной области.

Разработка динамических проектных процессов стала одним из новейших направлений в науке «Управление бизнес-процессами (BPM)». Исследователи и практики совершенствуют инструменты, методы и теорию гибкой разработки проектных процессов [1-4]. Идея гибкой разработки проектных процессов была взята в BPM из разработки программного обеспечения, где гибкая разработка программного обеспечения стала установленным термином и методом. Основные принципы гибкой разработки программного обеспечения изложены в манифесте [5]. Кроме того, манифест содержит большой объем практических исследований [6]. В работе [7] приводится базовое определение динамического потока работ при проектировании автоматизированных систем как потока проектных работ, приспособленного к изменениям в окружающей среде.

За последние 30 лет в области управления проектирования автоматизированных систем были решены важные проблемы создания, организации, анализа и контроля потоков работ, а также их реализации в практике разработки настоящих систем. В работе [8] дается подробный обзор методов и инструментов для организации потоков работ, которые включают следующие фазы: моделирование потоков работ «как есть», их оптимизация посредством создания модели процесса «чтобы быть», изменение организационной структуры, «прокрутка» потока работ,

усовершенствование процесса; обсуждается использование систем на этапах разработки потоков работ, таких как ERP-системы, Системы управления Технологического процесса (WfMS), инструменты моделирования процесса. Быстрая и динамическая разработка сложных автоматизированных систем связана с адаптацией потоков работ к изменениям в требованиях к системе. Под динамичностью предприятия принято понимать его свойство, обеспечивающее функционирование в динамическом мире [9], это касается двух способностей: 1) приспособиться к изменениям в окружающей среде; 2) обнаружить новые возможности, постоянно появляющиеся в динамическом мире для запуска абсолютно новых продуктов (служб). Становление динамичности требует разработку подхода, который позволяет обнаруживать изменения и воздействовать на них. Потребность в разработке такого подхода появилась как реакция на увеличение степени изменений в требованиях к разработке. Например, в работе [10] автор говорит: «требование изменений на этапах разработки при традиционных методах погружает успешность проекта в болото».

Whitestein Technologies, Magenta Technologies, SkodaAuto, Volkswagen, Saarstahl AG отмечают, что первое поколение статических (монолитных) систем управления жизненным циклом изделия и потоками проектных работ [11] больше не удовлетворяет потребности многих компаний, подход и автоматизированные средства первого поколения стандартизации процессов потоков проектных работ исчерпали свой ресурс, как следствие появляются плохо формализованные (неподходящие, содержащие семантические ошибки) процессы, стимулирующие рост затрат на их исправление, развитие и улучшение. При разработке автоматизированных систем, учитывающих время, с помощью подходов Scrum, LSPS [11] и других стратегическое соответствие (полнота, адекватность и непротиворечивость) проекта задуманному часто теряется

из-за большого объема информации. Чтобы проект соответствовал требованиям заказчика и был успешным, в помощь разработчику предлагается концептуальная модель, дающая общее представление о структуре организации проектного процесса с описанием акторов (сущностей) и их роли, функциях и связи друг с другом.

Анализом и контролем проектных потоков работ занимаются научные школы ГУВШЭ, МГТУ СТАНКИН, МВТУ им. Н.Э. Баумана, УлГТУ, ПОМИ АН РФ, ВМК МГУ имени М.В. Ломоносова, Института системного программирования РАН (Россия), университета Карнеги-Меллон (США), лаборатории VERIMAG (Франция), а также такие ученые как Афанасьев А. Н. [36-40], Карпов Ю. Г.[41], Соснин П. И., Лифшиц Ю.М. [42], Ярушкина Н. Г., Калянов Г. Н. [43], Конев Б. Ю., Шалыто А. А., Савенков К.О., Кулямин В. В., Охотин А.С., Михеев А.Г. (Россия), а также Neda Saeedloei, Gopal Gupta [44], Кларк Э.М., Буч Г.(США), Yuan Wang, Yushun Fan [34] (Китай), Van der Aalst.

Проектирование и обработка потоков работ связаны с технологией Rational Unified Process (RUP) [45], методологией PBWD, языками моделирования Unifeid Model Language (UML) [46], extended Event Driven Process Chain (eEPC), BPMN, IDEF0, IDEF3, Amber, Promela, YAWL, the Booch Methodology [47], Hierarchical Object Oriented Requirement Analysis (HOORA) [48], Jacobson Method [49], Object Modeling Technique (OMT) [50], Planguage [51], Shlaer-Mellor Object-Oriented Analysis Method [52], Software Cost Reduction requirements method (SCR) [53], Soft-ware Requirements Engineering Methodology (SREM) [54], Storyboard Proto-typing [55], Structured Analysis and Design Technique (SADT) [56], а также Structured Analysis and System Specification (SASS), Volere method, WinWin approach, Component-based methods (COTS-Aware Requirements Engineering (CARE), Off-the-Shelf Option (OTSO)) [57].

Карпов Ю. Г. в работе [41] использует подход Model Checking для анализа, контроля, моделирования и реинжиниринга бизнес-процессов, в котором главным недостатком является исследование модели, а не самой системы, поэтому возникает вопрос об адекватности модели к системе, при этом сложность решения перечисленных выше задач является экспоненциальной. В работе Neda Saeedloei and Gopal Gupta [44] применен темпоральный автомат, реализующий темпоральную контекстно-свободную грамматику, для анализа киберфизических систем с последующим переводом этой грамматики в программу для интерпретатора Prolog. Yuan Wang and Yushun Fan [34] предлагают использовать темпоральную логику действий для описания потоков работ в графовой форме, что требует описания всех маршрутов графа в формулах темпоральной логики действий. При этом применяется линейная темпоральная логика к формализации маршрута из задач, разветвлений AND, OR и схождений JOIN, однако вопрос адекватности построения описания потоков работ в графовой форме остается нерешенным.

База инструментальных средств анализа и контроля кибер-физических систем, а также потоков работ доступна по адресу [58, 64]. Кроме этого, имеются инструменты CPN Tools [59], «Roméo - A tool for Time Petri Nets analysis» [60], TimesTool [61], Tina Toolbox [62], Visual Object Net++ [63]. К традиционным система управления потоками работ относят ProBis [14]. К динамическим системам управления потоками проектных работ, согласно работам [15, 16, 17], относят YAWL (Yet Another Workflow Language), iPB. Во всех подобных системах используются диаграмматические представления потоков работ. При этом решаются задачи анализа структуры (синтаксиса) и смысла (семантики) диаграмм. Так, в работе [18] используются цветные сети Петри для динамического семантического анализа потоков работ, а в работе [19]

подход pi-Calculus, формализующий потоки работ в алгебраические высказывания логики первого порядка.

В настоящее время pi-исчисление (pi-calculus) является перспективной, но еще очень молодой и развивающейся теорией, в ней много открытых вопросов и нерешенных проблем. Сети Петри имеют следующие ограничения:

• нет универсального фреймворка для моделирования и анализа потоков проектных работ на базе сетей Петри. Для того, чтобы анализировать различные свойства (живость, достижимость, безопасность), потоки работ моделируются в разных типах сетей Петри, что является приемом Ad hoc [66];

• нет механизма, который помог бы проектировщику при моделировании и обеспечил успешное завершение задачи с необходимыми требованиями (свойствами).

Достаточно широкое применение анализа потоков работ при разработке безошибочных систем на этапе проектирования нашел метод model checking. Однако он предназначен для опытных ученых и инженеров, так как сложен в понимании и оперировании [34].

При проектировании и разработке автоматизированных систем (АС) (например, Web-систем в соответствии с архитектурой Service-Oriented Architecture) разработчики выделяют оркестровку (orchestration) и хореографию (choreography) в архитектуре системы [35]. Языками моделирования для описания оркестровки выступают BPEL (Business Process Executive Language), XPDL (XML Process Definition Language), UML (Unifided Model Language). Соответственно, языками моделирования для описания хореографии выступают WS-CDL (Web-services Choreography Description Language) и ebXML (Electronic Business using eXtensible Markup Language).

В современной теории графических визуальных языков для представления диаграмм используется логическая модель, содержащая графические объекты и связи между ними. Для обработки таких моделей используются графические грамматики. John L. Pfaltz и Azreil Rosenfeld предложили веб-грамматику [20]. Zhang и Costagliola [21, 22] разработали позиционную графическую грамматику, относящуюся к контекстно-свободной грамматике. Wittenberg и Weitzman [23] разработали реляционную графическую грамматику. Zhang и Orgun [24] описали сохраняющую графическую грамматику в своих работах. Однако упомянутые графические грамматики имеют следующие недостатки.

1. Позиционные грамматики, развиваясь на базе плекс-структур, не предполагают использование областей соединения и не могут применяться для графических языков, объекты которых имеют динамически изменяемое количество входов/выходов.

2. Авторы реляционных грамматик оговариваются о несовершенстве механизма нейтрализации в плане неполноты формируемого списка ошибок.

3. Отсутствует контроль семантической целостности (текстовой атрибутики комплексных диаграмматических моделей, представленных на различных визуальных языках), а также семантической согласованности в плане структурных вопросов диаграмм между собой и с концептуальной моделью в целом.

4. Общими недостатками вышеописанных грамматик являются: увеличение числа продукций при построении грамматики для неструктурированных графических языков (при неизменном количестве примитивов графического языка для описания всех вариантов неструктурированности происходит значительное увеличение количества продукций), сложность построения грамматики, большие временные

затраты анализа (анализаторы, построенные на базе рассмотренных грамматик, обладают полиномиальным или экспоненциальным временем анализа диаграмм графических).

В работах [25, 26] для обработки визуальных языков предлагается синтаксически-ориентированный подход на базе семейства RV-грамматик. Однако механизмы анализа и синтеза структурных и семантических особенностей диаграмм в плане их целостности и согласованности между собой и с концептуальной моделью, в том числе по текстовой атрибутике, отсутствуют.

Задача нейтрализации ошибок и ее решение хорошо отражены в классических работах по компиляторам, например, [27]. Предложен также метод нейтрализации ошибок для RV-грамматик [28]. Однако вопросы нейтрализации для диаграмматических моделей динамических распределенных потоков работ в них не решены.

Трансляция моделей визуального языка в другой целевой язык на основе RV-грамматик решается в работе [29]. Однако задача трансляции нескольких взаимосвязанных диаграмматических моделей потоков работ, представленных на различных языках, в целевой язык не рассматривается.

В наиболее распространенных инструментальных средствах создания и обработки диаграмматических моделей, таких как Microsoft Visio [30], Visual paradigm for UML [31], Aris Toolset [32], IBM Rational Software Architect (RSA) [33], анализ диаграмматических моделей производится прямыми методами, требует нескольких «проходов» в зависимости от контролируемого типа ошибки, отсутствует анализ структурных особенностей комплексных диаграмматических моделей, а также семантический анализ (денотативный и сигнификативный) диаграмматических моделей динамических потоков работ.

Актуальность задачи обучения проектировщиков анализу и

преобразованию потоков работ в условиях крупного радиотехнического предприятия обусловлена усложнением и появлением новых технических объектов, сокращением сроков их проектирования, повышением качества проектных решений. Появление новых САПР и множество развивающихся САПР требуют постоянного повышения квалификации проектировщиков предприятий. Важность корпоративного обучения подчеркивается правительством России, необходимость переподготовки специалистов является частью производственной политики современного предприятия и во многом определяет его интеллектуальный капитал и успешность на рынке.

Компьютерные технологии проектирования являются основными в производственном цикле, предлагается множество отечественных и зарубежных САПР различного назначения, класса и стоимости. Однако в области обучения автоматизированному проектированию промышленных объектов имеется ряд нерешенных проблем.

1. Отсутствуют эффективные средства адаптации обучаемого проектировщика к учебно-практическому наполнению в ходе процесса обучения, позволяющие сократить сроки обучения без отрыва от производства.

2. Автоматизированные обучающие системы (АОС) САПР являются статическими с заранее заданной неизменной структурой, ориентированы на целевую аудиторию с максимальной степенью усвоения материала, в них не учитываются динамические индивидуальные характеристики обучаемых.

3. В основе реализации АОС САПР лежит, как правило, модульный принцип, что снижает степень масштабируемости архитектуры.

4. АОС САПР являются узко специализированными в предметной области.

В работе представлены новые методы, модели и средства анализа и преобразования диаграмматических моделей гибридных динамических потоков проектных работ для решения вышеуказанных проблем, а также модели и метод адаптивного управления процессом обучения.

Глава 1. Диаграмматика потоков проектных работ в автоматизации проектирования

1. Научно-техническая проблема, имеющая важное хозяйственное значение

Фундаментальной научной проблемой является повышение эффективности обработки диаграмматических моделей потоков проектных работ автоматизированных систем с целью сокращения временных затрат на их разработку, повышение успешности обработки диаграмматических моделей потоков проектных работ, а именно, выполнение требования к ресурсным ограничениям, функционалу, финансовой составляющей и срокам исполнения, а также повышение качества диаграмматических моделей в плане контроля ошибок, сужения семантического разрыва между анализом бизнес-процессов и их выполнением. Проблема в техническом плане представлена в работе [11]. Данная работа вносит вклад в решение научно-технической проблемы новым грамматико-алгебраическим подходом к контролю и анализу денотативных и сигнификативных семантических ошибок диаграмматических моделей потоков проектных работ, приблизив решение к оптимуму. Поскольку согласно работе [41] невозможно формально определить, что означает «полное отсутствие ошибок» в диаграмматических моделях потоков проектных работ, то под оптимумом понимается контроль и анализ всех известных ошибок динамических моделей потоков работ, включая синтаксические (структурные), семантические (денотативные, сигнификативные и квантовые), а также контроль топологических событий, имен и т.д. (в работе представлено 23 класса ошибок). В работах [34, 43, 44] рассматриваются подходы и методы контроля, анализа и моделирования бизнес-процессов.

2. Разработка и исследование синтаксически-ориентированных методов контроля и анализа диаграммных языков потоков проектных работ

Целью настоящего параграфа является разработка синтаксически-ориентированного метода анализа и контроля диаграммных языков потоков проектных работ (ППР). Изложение материала ведется на примере исследований ППР в диаграмматике параллельных графических схем (ПГС) [67], в которой присутствуют все синтаксические и семантические особенности диаграммных моделей ППР.

RV-грамматика является развитием и расширением R, RS и RG-грамматик [69, 70] и предназначена для синтаксического и семантического анализа и контроля диаграммных языков.

Определение 1. RV-грамматикой [71, 72, 73, 74, 75, 76] языка *L(G)* называется упорядоченная пятерка непустых множеств $G = (V, \sum, \tilde{\sum}, R, r_0)$, где

- $G = (V, \sum, \tilde{\sum}, R, r_0)$, $V = \{v_e, e = \overline{1, L}\}$ – вспомогательный алфавит (алфавит операций над внутренней памятью);
- $\Sigma = \{a_t, t = \overline{1, T}\}$ – терминальный алфавит графического языка, являющийся объединением множеств его графических объектов и связей (множество примитивов графического языка);
- $\tilde{\Sigma} = \{\tilde{a}_t, t = \overline{1, \tilde{T}}\}$ – квазитерминальный алфавит, являющийся расширением терминального алфавита.

Алфавит $\tilde{\Sigma}$ включает:

– квазитермы графических объектов с одним входом;

– квазитермы графических объектов, имеющих более одного входа;

– квазитермы связей-меток с определенными для них семантическими различиями. Связью-меткой является каждая связь, исходящая из графического объекта, содержащего более одного входа или

выхода. Допустимо одну из связей такого графического объекта не обозначать как связь-метка. Механизм меток используется для возврата к непроанализированным частям графического образа;

– квазитерм, определяющий отсутствие связей-меток. Квазитерм no_label, определяющий отсутствие связей-меток, используется для прекращения анализа диаграммы при условии выполнения операции (ий) над внутренней памятью.

• $R = \{r_i, i = \overline{0,I}\}$ – схема грамматики G (множество имен комплексов продукций, причем каждый комплекс r_i состоит из подмножества P_{ij} продукций $r_i = \{P_{ij}, j = \overline{1,J}\}$);

• $r_0 \in R$ – аксиома RV-грамматики (имя начального комплекса продукций), $r_k \in R$ – заключительный комплекс продукций.

• Продукция $P_{ij} \in r_i$ имеет вид $P_{ij} : \tilde{a}_t \xrightarrow{\Omega_\mu\left[W_\nu\left(\gamma_1,..,\gamma_n\right)\right]} r_m$, где $W_\nu\left(\gamma_1,...,\gamma_n\right)$ – n-арное отношение, определяющее вид операции над внутренней памятью в зависимости от $\nu \in \{0,1,2,3\}$;

• Ω_μ – оператор модификации определенным образом изменяющий вид операции над памятью, причем $\mu \in \{0,1,2\}$;

• $r_m \in R$ – имя комплекса продукции-преемника.

В качестве внутренней памяти предлагается использовать стеки и магазины для обработки графических объектов имеющих более одного выхода (чтобы хранить информации о связях-метках) и эластичные ленты [77] для обработки графических объектов, имеющих более одного входа (чтобы отмечать количество возвратов к данной вершине, а, следовательно, количество входящих дуг). Ячейки лент могут работать в режиме счетчика целых положительных чисел.

Определим следующую интерпретацию правил RV-грамматики в зависимости от параметров μ и ν.

Пусть $\mu = 0$, тогда $\Omega_0\left[W_\nu\left(\gamma_1, ..., \gamma_n\right)\right] \equiv W_\lambda\left(\gamma_1, ..., \gamma_n\right)$, т. е. Ω_0 – пустой оператор.

Отношение $W_\lambda\left(\gamma_1, ..., \gamma_n\right)$ определяет операции над памятью, реализованной моделями стек или магазин.

При $\lambda = 0$ никаких действий над памятью не производится.

При $\lambda = 1$ ($\lambda = 2$) в стек / магазин с номером $S \in \{\overline{1,n}\}$ записывается (стирается) $\gamma_S \in V$, причем если запись производится безусловно, то стирание осуществляется при условии, что γ_S в правиле RV-грамматики и вершине стека/магазины совпадают. В противном случае данное правило считается неприменимым (стирание не производится).

При $\mu = 1$ оператор Ω_1 имеет вид

$$\Omega_1\left[W_\nu\left(\gamma_1, ..., \gamma_n\right)\right] \equiv W_\lambda\left(\gamma_1^{\alpha_1}, ..., \gamma_n^{\alpha_n}\right),$$

определен для $\lambda = 1,2,3$ и задает простую операцию записи, чтения или сравнения над ленточной памятью, где $(\alpha_1(L_1), ..., \alpha_n(L_n))$ указывают номера ячеек соответствующих эластичных лент $(1, ..., n)$, куда будут записаны при $\lambda = 1$ (считаны при $\lambda = 2$, сравнены при $\lambda = 3$) символы $\gamma_1, ..., \gamma_n \in V$.

При $\mu = 2$ оператор Ω_2 имеет вид

$$\Omega_2\left[W_\nu\left(\gamma_1, ..., \gamma_n\right)\right] \equiv W_{\lambda_1}\left(\gamma_1^{\alpha_1}, ..., \gamma_n^{\alpha_n}\right) / W_{\lambda_2}\left(\gamma_1^{\beta_1}, ..., \gamma_n^{\beta_n}\right),$$

определен для $\lambda_1 = 1,2$, $\lambda_2 = 2,3$ и задает условную операцию над ленточной памятью, т. е. операция в числителе выполняется при условии выполнения операции в знаменателе.

В операциях над внутренней памятью номера ячеек лент (α_i) и значения в магазинах/стеках кодируются следующим образом:

• t – номер текущего графического примитива из числа примитивов данного типа;

• b – номер графического примитива, от которого исходит «управляющий сигнал» (применяется только для связей).

Цепочка $\varphi = \tilde{a}_{t_1}, \tilde{a}_{t_2} \ldots, \tilde{a}_{t_\lambda}$ называется RV-выводом $\tilde{a}_{t_\lambda}$ из $\tilde{a}_{t_1}$ и обозначается $\tilde{a}_{t_1} \overset{RV}{\Longrightarrow} \tilde{a}_{t_\lambda}$, если для любого $\xi < \lambda$ существует r_i и $r_e \in R$ такие, что $\tilde{a}_{t_{\zeta+1}} \in r_e$, $(\tilde{a}_t \xrightarrow{\Omega_\mu \left[W_\nu (\gamma_1, \ldots, \gamma_n) \right]} r_e) \in r_i$. RV-вывод называется законченным (обозначается $\tilde{a}_{t_1} \overset{RV}{\Longrightarrow} \tilde{a}_{t_\lambda}$), если $\tilde{a}_{t_\lambda}$ порождается продукцией с r_k в правой части.

RV-грамматика эффективна как для порождения, так и для распознавания.

Порождение некоторой цепочки языка L по его RV-грамматике G начинается с применения любой из продукций комплекса r_0. Эта продукция определяет начальный символ порождаемой цепочки, а кроме того, она определяет необходимую операцию над внутренней памятью и имя применимого комплекса продукций-преемников, из которых выбирается очередная продукция. Порождение заканчивается применением продукции с r_k в правой части.

Распознавание принадлежности некоторой цепочки языка L, заданному RV-грамматикой, сводится к проверке вхождения первого символа проверяемой цепочки левой части какой-либо продукции комплекса r_0, в то время как последующие символы должны встречаться в левой части теку-щего комплекса продукций-преемников, а последний символ цепочки обязательно должен принадлежать продукции с r_k в левой части.

Применение каждой продукции обязательно должно сопровождаться соответствующими действиями над внутренней памятью. В начале процесса порождения и распознавания внутренняя память должна быть

пуста, а по окончанию – состояние памяти должно проверяться операциями продукции с r_k в правой части.

Построение RV-грамматики реализуется в две фазы: синтез и анализ.

Синтез RV-грамматики состоит из следующих этапов.

1. Определяется терминальный алфавит контролируемого графического языка, описывается расположение меток, выявляются семантические различия для связей, имеющих общее графическое представление, строится алфавит квазитермов.

2. Строится матрица допустимых паросочетаний для квазитерминаль-ного алфавита.

3. Определяются отношения над внутренней памятью, обеспечивающие эффективный контроль связности графических объектов.

4. По матрице допустимых паросочетаний, т. е. системе отношений, строится граф метаязыка RV-грамматик, вершинам которого поставлены в соответствие имена комплексов правил, а дугам – квазитермы и операции над внутренней памятью. Помимо графовой метаязык RV-грамматики может быть представлен также в табличной и аналитической формах.

Анализ осуществляется в два этапа.

1. Устраняются недетерминированности и неопределенности.

2. Производится минимизация RV-грамматики.

Ниже приведен пример построения RV-грамматики для языка ПГС.

Терминальный алфавит для языка ПГС представлен на рис. 1. Формируя квазитерминальный алфавит, необходимо выделить семантические различия (квазитермы) для связей. Для языка ПГС таковых будет пять: одна из связей, исходящих из графического объекта «логическое условие»; связь, исходящая из графического объекта

«объединение взаимоисключающих ветвей»; все, кроме одной, связи, исходящие из графического объекта «распараллеливание»; связь, исходящая из графи-ческого объекта «слияние»; все остальные связи, не вошедшие в предыдущие пункты перечня.

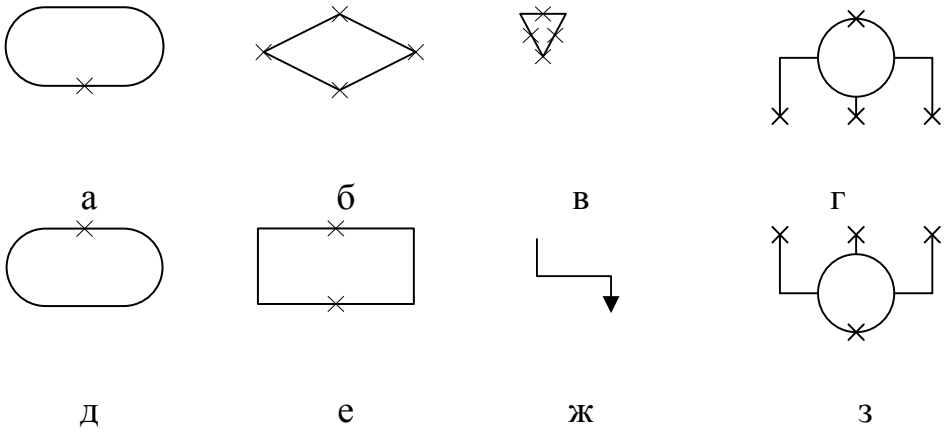
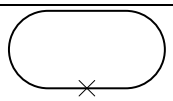
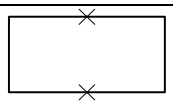
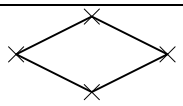


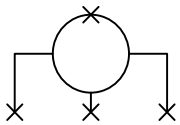
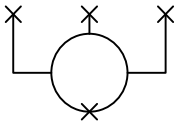
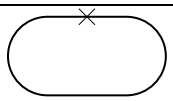
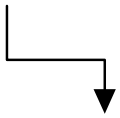
Рис. 1. Графические примитивы языка параллельных графических схем алгоритма (ПГСА) (а – начало, б – логическое условие, в – объединение взаимоисключающих ветвей, г – распараллеливание, д – конец, е – действие, ж – связь, з – слияние)

Кроме этого, объединение взаимоисключающих ветвей и слияние также наделяются дополнительным квазитермом, так как имеют более одного входа. В результате получаем квазитерминальный алфавит, представленный в таблице 1.

Таблица 1. Терминальный и квазитерминальный алфавиты языка ПГС для RV-грамматики

Алфавит термов	**Алфавит квазитермов**	**Примечание**
	A_0	
	A	
	P	

Окончание табл. 1

Алфавит термов	**Алфавит квазитермов**	**Примечание**
	W	
	$\underline{W}$	уже анализированное объединение заимоисключающих ветвей
	R	
	L	
	$\underline{L}$	уже анализированное слияние
	A_k	
	*rel*	
	$label_P$	связь-метка, исходящая из логического условия
	$label_W$	связь-метка, исходящая из объединения взаимоисключающих ветвей
	$label_R$	связь-метка, исходящая из распараллеливания
	$label_L$	связь-метка, исходящая из слияния
	no_label	отсутствие связей

Отношения над внутренней памятью определяются для продукций, содержащих:

- квазитермы связей-меток (в нашем случае $label_P$, $label_W$, $label_R$ и $label_L$);

- квазитермы отсутствия связей-меток (no_label);
- квазитермы графических объектов, имеющих более одного входа или выхода ($.P.$, W, R и L);
- дополнительные квазитермы графических объектов, имеющих более одного входа ($\underline{W}$ и $\underline{L}$).

В качестве внутренней памяти будем использовать:

- четыре магазина для хранения информации для связей-меток исходящих из логического условия (1), объединения взаимоисключающих ветвей (2), распараллеливания (3) и слияния (4);
- три ленты для хранения информации об уже проанализированных объединениях взаимоисключающих ветвей (1), слияниях (2) и количестве входов в слияния (3).

Отношение вида $W_2(t^{1m})$, определенное для логического условия P, осуществляет запись ссылки на связь-метку в первый магазин, т. е. устанавливает точку возврата.

При анализе распараллеливания основной задачей является сохранение связей меток, поэтому операции над внутренней памятью выглядят следующим образом $W_1(t^{3m^{(k-1)}}) / W_3(k>1)$. В третий магазин заносятся ссылки на связи-метки, исходящие из распараллеливания, в количестве на единицу меньшем общего количества исходящих связей из текущего элемента.

Анализатор, производя разбор, дважды достигает объединения взаимоисключающих ветвей в W и $\underline{W}$. В первом случае применяется операция вида $W_1(1^{t(1)}, t^{2m}) / W_2(e^{t(1)})$, которая означает запись 1 в ячейку с номером t ленты (отмечает первое прохождение объединения взаимоисключающих ветвей) и запись номера текущего элемента во второй магазин (устанавливает точку возврата) при условии, что ячейка с

номером t ленты пуста (анализатор не проходил данное объединение взаимоисклю-чающих ветвей). Во втором случае операция вида $W_1(2^{t(1)}) / W_2(1^{t(1)})$, которая означает запись 2 в ячейку с номером t ленты (отмечает второе прохождение объединения взаимоисключающих ветвей) при условии, что в ячейке с номером t ленты 1 (анализатор однажды проходил данное объединение взаимоисключающих ветвей).

По схожему алгоритму происходит анализ слияния, представленного двумя квазитермами L и $\underline{L}$. Для первого из них используется операция вида $W_1(1^{t(2)}, k^{t(3)}, t^{4m}) / W_2(e^{t(2)})$, означающая запись 1 в ячейку с номером t второй ленты (отмечает первое прохождение объединения взаимоисключающих ветвей), количества входящих связей в ячейку t третей ленты и в четвертый магазин – ссылки на связь-метку, исходящую из слияния, при условии его анализа впервые. Для второго назначается операция вида $W_1(inc(m^{t(2)})) / W_3(m^{t(2)} < k^{t(3)})$ – увеличение значения в ячейке t второй ленты при условии, что анализ этого слияния может быть продолжен (количество проанализированных входящих связей меньше их общего количества).

Для связей-меток $label_P$, $label_W$, $label_R$ определены отношение вида $W_2(b^{1m})$, $W_2(b^{2m})$, $W_2(b^{3m})$, соответственно, они предполагают извлечение из магазина ссылки на связь, по которой нужно следовать.

Операция вида $W_2(b^{4m}) / W_3(m^{t(2)} == k^{t(3)})$ назначается для связи-метки $label_L$ и предполагает извлечение из четвертого магазина ссылки на связь, по которой нужно следовать при условии, что все связи, входящие в слияние, пройдены.

Символ «$*$» обозначает операции вида:

$$* = W_2(e^{1m}) \,\&\&\, W_2(e^{2m}) \,\&\&\, W_2(e^{3m}) \,\&\&\, W_2(e^{4m}) \,\&\&\, W_2(2^{\alpha_i(1)}) \,\&\&\, W_3(m^{\alpha_i(2)} > 1) \,\&\&\, W_3(m^{\alpha_i(2)} == k^{\alpha_i(3)})$$

Причем операции $W_2(e^{1m})$, $W_2(e^{2m})$, $W_2(e^{3m})$ и $W_2(e^{4m})$ означают, что все

магазины должны быть пусты (т. е., нет точек возврата), операция $W_2(2^{\alpha_i(1)})$ – чтение единиц из всех ячеек ленты, а операции $W_3(m^{\alpha_i(2)} > 1) \,\&\,\& \, W_3(m^{\alpha_i(2)} == k^{\alpha_i(3)})$ – проверку количества входящих связей, что их больше единицы и идентично общему количеству входов соответствующего элемента.

Введение операций над памятью достаточно для обеспечения эффективного контроля связности графических объектов. Так, операции $W_1(1^{t(1)}, t^{2m}) / W_2(e^{t(1)})$ и $W_1(2^{t(1)}) / W_2(1^{t(1)})$ не допускают трех и более кратный проход анализатора через объединение взаимоисключающих ветвей. Операция «$*$« выполняется после анализа последнего символа. Она обеспечивает обнаружение ошибок следующих типов: отсутствие второй связи, исходящей из логического условия, отсутствие входящих (исходящих) связей в (из) объединение взаимоисключающих ветвей и т. п.

Построение табличной формы RV-грамматики производится построчным просмотром матрицы допустимых паросочетаний. Для каждой встреченной «1» осуществляется запись значения первого столбца данной строки и значений первой и второй строки данного столбца, т. е. имя комплекса, имя комплекса-преемника и квазитерм, соответственно. После чего к каждой продукции приписываются ранее определенные операции над внутренней памятью. RV-грамматика ПГС приведена в таблице 2.

Таблица 2. RV-грамматика ПГС

№	Комплекс-источник	Квази-терм	Комплекс-приемник	RV-отношение
1	r_0	A_0	r_1	∅
2	r_1	*rel*	r_9	∅
3	r_2	*rel*	r_9	∅
4	r_3	*rel*	r_9	∅

Продолжение табл. 2

№	Комплекс-источник	Квази-терм	Комплекс-приемник	RV-отношение
5	r_4	$label_P$	r_{10}	$W_2(b^{1m})$
6		$label_W$	r_{11}	$W_2(b^{2m})$
7		$label_R$	r_{12}	$W_2(b^{3m})$
8		$label_L$	r_{13}	$W_2(b^{4m}) / W_3(m^{t(2)} == k^{t(3)})$
9	r_5	$label_P$	r_{10}	$W_2(b^{1m})$
10		$label_W$	r_{11}	$W_2(b^{2m})$
11		$label_R$	r_{12}	$W_2(b^{3m})$
12		$label_L$	r_{13}	$W_2(b^{4m}) / W_3(m^{t(2)} == k^{t(3)})$
13		no_label	r_k	*
14	r_6	rel	r_9	
15	r_7	$label_P$	r_{10}	$W_2(b^{1m})$
16		$label_W$	r_{11}	$W_2(b^{2m})$
17		$label_R$	r_{12}	$W_2(b^{3m})$
18		$label_L$	r_{13}	$W_2(b^{4m}) / W_3(m^{t(2)} == k^{t(3)})$
19	r_8	$label_P$	r_{10}	$W_2(b^{1m})$
20		$label_W$	r_{11}	$W_2(b^{2m})$
21		$label_R$	r_{12}	$W_2(b^{3m})$
22		$label_L$	r_{13}	$W_2(b^{4m}) / W_3(m^{t(2)} == k^{t(3)})$
23		no_label	r_k	*
24	r_9	A	r_2	$\emptyset$

Продолжение табл. 2

№	Комплекс-источник	Квази-терм	Комплекс-приемник	RV-отношение
25		P	r_3	$W_1(t^{1m})$
26	r_9	W	r_4	$W_1(1^{t(1)}, t^{2m}) / W_2(e^{t(1)})$
27		$\breve{W}$	r_5	$W_1(2^{t(1)}) / W_2(1^{t(1)})$
28		R	r_6	$W_1(t^{3m^{(k-1)}}) / W_3(k > 1)$
29		L	r_7	$W_1(1^{t(2)}, k^{t(3)}, t^{4m}) / W_2(e^{t(2)})$
30		$\breve{L}$	r_8	$W_1(inc(m^{t(2)})) / W_3(m^{t(2)} < k^{t(3)})$
31		A_k	r_{14}	$\emptyset$
32	r_{10}	A	r_2	$\emptyset$
33		P	r_3	$W_1(t^{1m})$
34		W	r_4	$W_1(1^{t(1)}, t^{2m}) / W_2(e^{t(1)})$
35		$\breve{W}$	r_5	$W_1(2^{t(1)}) / W_2(1^{t(1)})$
36		R	r_6	$W_1(t^{3m^{(k-1)}}) / W_3(k > 1)$
37		L	r_7	$W_1(1^{t(2)}, k^{t(3)}, t^{4m}) / W_2(e^{t(2)})$
38		$\breve{L}$	r_8	$W_1(inc(m^{t(2)})) / W_3(m^{t(2)} < k^{t(3)})$
39		A_k	r_{14}	$\emptyset$
40	r_{11}	A	r_2	$\emptyset$
41		P	r_3	$W_1(t^{1m})$
42		W	r_4	$W_1(1^{t(1)}, t^{2m}) / W_2(e^{t(1)})$
43		$\breve{W}$	r_5	$W_1(2^{t(1)}) / W_2(1^{t(1)})$
44		R	r_6	$W_1(t^{3m^{(k-1)}}) / W_3(k > 1)$

Продолжение табл. 2

№	Комплекс-источник	Квази-терм	Комплекс-приемник	RV-отношение
45		L	r_7	$W_1(1^{t(2)}, k^{t(3)}, t^{4m}) / W_2(e^{t(2)})$
46		$\breve{L}$	r_8	$W_1(inc(m^{t(2)})) / W_3(m^{t(2)} < k^{t(3)})$
47		A_k	n_{14}	$\oslash$
48	n_{12}	A	r_2	$\oslash$
49		P	r_3	$W_1(t^{1m})$
50		W	r_4	$W_1(1^{t(1)}, t^{2m}) / W_2(e^{t(1)})$
51		$\breve{W}$	r_5	$W_1(2^{t(1)}) / W_2(1^{t(1)})$
52	n_{12}	R	r_6	$W_1(t^{3m^{(k-1)}}) / W_3(k > 1)$
53		L	r_7	$W_1(1^{t(2)}, k^{t(3)}, t^{4m}) / W_2(e^{t(2)})$
54		$\breve{L}$	r_8	$W_1(inc(m^{t(2)})) / W_3(m^{t(2)} < k^{t(3)})$
55		A_k	n_{14}	$\emptyset$
56	n_{13}	A	r_2	$\emptyset$
57		P	r_3	$W_1(t^{1m})$
58		W	r_4	$W_1(1^{t(1)}, t^{2m}) / W_2(e^{t(1)})$
59		$\breve{W}$	r_5	$W_1(2^{t(1)}) / W_2(1^{t(1)})$
60		R	r_6	$W_1(t^{3m^{(k-1)}}) / W_3(k > 1)$
61		L	r_7	$W_1(1^{t(2)}, k^{t(3)}, t^{4m}) / W_2(e^{t(2)})$
62		$\breve{L}$	r_8	$W_1(inc(m^{t(2)})) / W_3(m^{t(2)} < k^{t(3)})$
63		A_k	n_{14}	$\oslash$
64	n_{14}	$label_P$	n_{10}	$W_2(b^{1m})$
65		$label_W$	n_{11}	$W_2(b^{2m})$

Окончание табл. 2

№	Комплекс-источник	Квази-терм	Комплекс-приемник	RV-отношение
66		$label_R$	r_{12}	$W_2(b^{3m})$
67		$label_L$	r_{13}	$W_2(b^{4m}) / W_3(m^{t(2)} == k^{t(3)})$
68		no_label	r_k	*
69	r_k			$\emptyset$

Процедура устранения недетерминированности начинается с определения дуг в графе (см. табл. 2), исходящих из одной и той же вершины, входящих в различные и содержащих одинаковые квазитермы. Неопределенность (неоднозначность) вносят дуги графа (продукции), входы (имя комплекса источника) и выходы (имя комплекса преемника) которых соответственно одинаковы, причем они нагружены различными операциями над внутренней памятью и одинаковыми квазитермами. Для устранения недетерминированности и неопределенности в графе необходимо предусмотреть более сложные операции над памятью, которые бы учитывали некоторую предысторию. Данный пример детерминирован. Неопределенность устранена для квазитермов W и $\underline{W}$ введением проверки $W_2(e^{t(1)})$ и $W_2(1^{t(1)})$.

Задача минимизации RV-грамматики имеет целью построение эквивалентной грамматики, содержащей меньшее количество продукций.

Задача минимизации решается без модификации контролируемого языка за счет приведения эквивалентных комплексов и введения пустых продукций.

Анализируя таблицу 2, находим группы эквивалентных состояний распознающего автомата (r_1, r_2, r_3, r_6), (r_4, r_7), (r_5, r_8, r_{14}), $(r_9, r_{10}, r_{11}, r_{12}, r_{13})$. Эти комплексы эквивалентны, так как допускают в точности одинаковые

продолжения распознаваемой цепочки справа. Группы (r_4, r_7) и (r_5, r_8, r_{14}) можно объединить в одну, поскольку первая поглощается второй, и продук-ции, присутствующие во второй группе, но отсутствующие в первой, не наносят качественного вреда анализатору.

Процедура минимизации RV-грамматики за счет приведения эквивалентных комплексов состоит в выполнении следующих действий:

- выделение эквивалентных комплексов продукций и выбор по одному представителю класса эквивалентности (в нашем случае выбираем r_1, r_5 и r_9);
- исключение всех остальных комплексов продукций, принадлежащих данному классу эквивалентности, кроме выбранных представителей;
- коррекция RV-грамматики, которая заключается в том, что в правых частях всех оставшихся продукций имена исключенных комплексов заменяются именами представителей классов эквивалентности.

В данном случае указанная процедура позволила построить RV-таблицу, содержащую 16 строк, вместо 69, т. е. получена экономия в 75%. Результат минимизации представлен в табличной форме в таблице 3, в аналитической форме на рис. 2 и графовой форме на рис. 3.

Таблица 3. Табличная форма RV-грамматики для языка ПГС после минимизации

№	Комплекс-источник	Квази-терм	Комплекс-приемник	RV-отношение
1	r_0	A_0	r_1	∅
2	r_1	rel	r_3	∅
3	r_2	$label_P$	r_3	$W_2(b^{1m})$
4		$label_W$	r_3	$W_2(b^{2m})$
5		$label_R$	r_3	$W_2(b^{3m})$
6		$label_L$	r_3	$W_2(b^{4m}) / W_3(m^{t(2)} == k^{t(3)})$

Окончание табл. 3

№	Комплекс-источник	Квази-терм	Комплекс-приемник	RV-отношение
7		no_label	r_k	*
8	r_3	A	r_1	$\oslash$
9		P	r_1	$W_1(t^{1m})$
10		W	r_2	$W_1(1^{t(1)}, t^{2m}) / W_2(e^{t(1)})$
11		$\breve{W}$	r_2	$W_1(2^{t(1)}) / W_2(1^{t(1)})$
12		R	r_1	$W_1(t^{3m^{(k-1)}}) / W_3(k > 1)$
13		L	r_2	$W_1(1^{t(2)}, k^{t(3)}, t^{4m}) / W_2(e^{t(2)})$
14		$\breve{L}$	r_2	$W_1(inc(m^{t(2)})) / W_3(m^{t(2)} < k^{t(3)})$
15		A_k	r_2	$\emptyset$
16	r_k			$\emptyset$

$G = (V, \Sigma, \widetilde{\Sigma}, R, r_0)$,

$V = \{1, 2, 3\}$,

$\Sigma = \{A_0, A, P, W, R, L, A_k, rel\}$,

$\widetilde{\Sigma} = \{A_0, A, P, W, \underline{W}, R, L, \underline{L}, A_k, rel, label_P, label_W, label_R, label_L, no\_label\}$,

$R = \{r_0, r_1, r_2, r_3, r_k\}$,

$r_0 = \{A_0 1\}$,

$r_1 = \{rel 3\}$,

$r_2 = \{label_P W_2(b^{1m})3, label_W W_2(b^{2m})3, label_R W_2(b^{3m})3,$

$label_L W_2(b^{4m}) / W_3(m^{t(2)} == k^{t(3)})3, no\_label * k\}$,

$r_3 = \{A1, P W_1(t^{1m})1, W W_1(1^{t(1)}, t^{2m}) / W_2(e^{t(1)})2, \underline{W} W_1(2^{t(1)}) / W_2(1^{t(1)})2,$

$R W_1(t^{3m^{(k-1)}}) / W_3(k > 1)1, L W_1(1^{t(2)}, k^{t(3)}, t^{4m}) / W_2(e^{t(2)})2,$

$\underline{L} W_1(inc(m^{t(2)})) / W_3(m^{t(2)} < k^{t(3)})2, A_k 2\}$,

$r_k = \{\oslash\}$.

Рис. 2. Аналитическая форма RV-грамматики для языка ПГС после минимизации

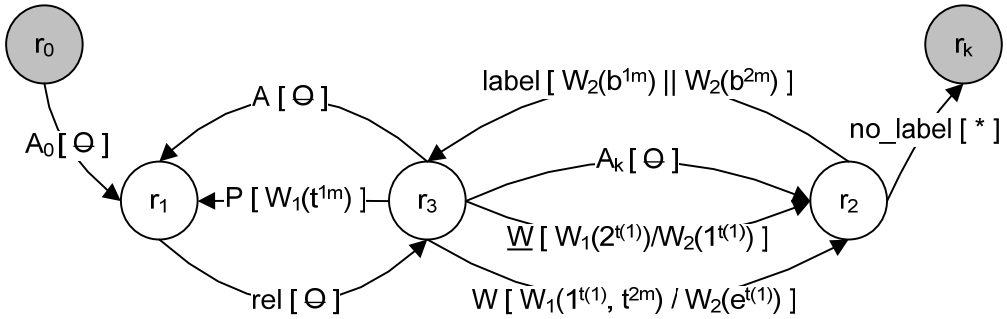


Рис. 3. Графовая форма RV-грамматики для языка ПГС после минимизации

На рис. 4 приведена структура линейно-ограниченного автомата, соответствующего RV-грамматике языка ПГС. Заштрихованные головки позволяют производить запись и чтение. Заштрихованные наполовину – только чтение информации.

Ленты L_1, L_2, L_3 хранят информацию об уже проанализированных объединениях взаимоисключающих ветвей, слияниях и количестве входов в слияния, соответственно. Очереди M_1, M_2, M_3, M_3 хранят информацию о связях-метках, исходящих из логического условия, объединения взаимоисключающих ветвей, распараллеливания и слияния, соответственно (для наглядности имеет смысл показывать, откуда исходит связь).

КУ – конечное устройство управления или функциональная схема автомата, в которой «зашит» алгоритм контроля, представленный графом переходов автомата. Элементы входного списка расположены неупорядоченно, порядок перехода к тому или иному элементу списка определяется конечным устройством анализатора.

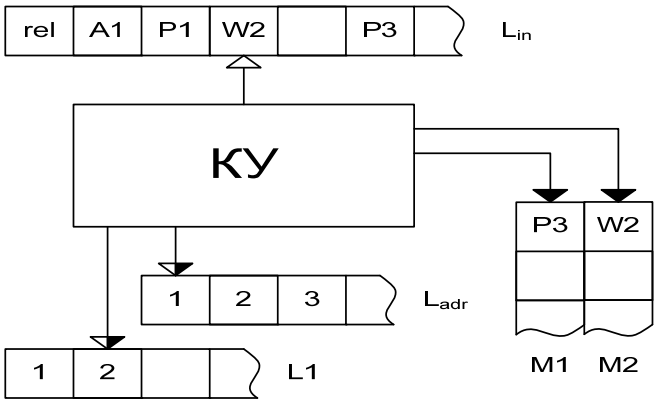


Рис. 4. RV-анализатор

Несмотря на большой интерес исследователей к графическим языкам и средствам их анализа, проблеме нейтрализации синтаксических и семантических ошибок уделяется мало внимания. В то же время, эффективные технологии обработки ошибок необходимы для анализаторов графических языков, для обеспечения возможности обнаружения как можно большего количества ошибок за один проход.

Предлагается формализм автоматной графической грамматики, способной выполнить нейтрализацию ошибок (RVN-грамматика), являющейся расширением RV-грамматики. Анализатор, построенный на базе этой грамматики, позволяет выявлять более одной ошибки за проход и обеспечивает линейное время анализа.

Определение 2. RVN-грамматикой [78, 79] языка *L(G)* называется упорядоченная пятерка непустых множеств $G = (V, \Sigma, \tilde{\Sigma}, R, n_0)$, где

- $V = \{\nu_e, e = \overline{1, L}\}$ – вспомогательный алфавит (алфавит операций над внутренней памятью);
- $\Sigma = \{a_t, t = \overline{1, T}\}$ – терминальный алфавит графического языка, являющийся объединением множеств его графических объектов и связей (множество примитивов графического языка);
- $\tilde{\Sigma} = \{\tilde{a}_t, t = \overline{1, \tilde{T}}\}$ – квазитерминальный алфавит, являющийся расширением терминального алфавита. Алфавит $\tilde{\Sigma}$ включает:

– квазитермы графических объектов, не являющихся продолжателями анализа;

– квазитермы графических объектов-продолжателей анализа;

– квазитермы графических объектов, имеющих более одного входа;

– квазитермы связей-меток с определенными для них семантическими различиями;

– квазитерм для проверки наличия графических объектов-продолжателей анализа;

– квазитерм для завершения анализа.

- $R = \{r_i, i = \overline{0,I}\}$ – схема грамматики G (множество имен комплексов продукций, причем каждый комплекс r_i состоит из подмножества P_{ij} продукций $r_i = \{P_{ij}, j = \overline{1,J}\}$);

- $r_0 \in R$ – аксиома RVN-грамматики (имя начального комплекса продукций), $r_k \in R$ – заключительный комплекс продукций.

Для нейтрализации ошибок на этапе проектирования грамматики выделяются типы графических объектов, содержащих более одного входа или выхода, экземпляры которых будут использоваться в качестве продолжателей анализа. Такие графические объекты являются «ключевыми», так как на них строятся основные графические конструкции. Кроме этого, за счет большого количества исходящих связей они позволяют охватить большую часть диаграммы для анализа.

Квазитермы графических объектов-продолжателей анализа необходимы для обработки следующих событий: «первичный» анализ объекта, «вторичный» анализ объекта.

Эти события возникают вследствие следующих типов переходов к графическому объекту во время анализа: естественного перехода от связи к графическому объекту; выбор данного объекта в качестве продолжателя анализа.

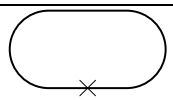
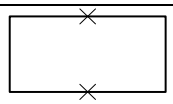
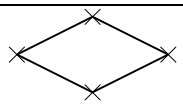
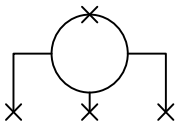
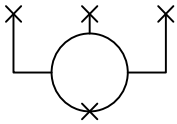
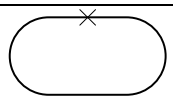
Работа с внутренней памятью, методика построения RVN-грамматики практически аналогична RV-грамматике. Единственным отличием является добавление элементов памяти для хранения информации о продолжателях (магазин для списка возможных продолжателей и лента для проанализированных графических объектов).

Ниже приведен пример RVN-грамматики для языка ПГС.

Терминальный алфавит для языка ПГС представлен в таблице 4. Определив семантические различия для связей, выбрав продолжателей

анализа (условие, объединение взаимоисключающих ветвей, распараллеливание, слияние), получаем квазитерминальный алфавит.

Таблица 4. Терминальный и квазитерминальный алфавиты языка ПГС для RVN-грамматики

Алфавит термов	**Алфавит квазитермов**
	A_0
	A
	P_a, P_b
	W_a, W_b, $\underline{W}$
	R_a, R_b
	L_a, L_b, L
	A_k
	rel, $label_P$, $label_W$, $label_R$, $label_L$
	$find$, no_label

Значения квазитермов:

- $find$ – поиск продолжателей анализа;
- no_label – завершения анализа;
- для элемента $E = \{P, W, R, L\}$, $label_E$ – связь-метка, исходящая из E, E_a – первичный анализ E, E_b – вторичный анализ E, E – уже анализированное E.

Далее, согласно методике построение RVN-грамматики определяем

операции над внутренней памятью, допустимые паросочетания, производим построение грамматики и минимизируем ее. Результат этих действий представлен в таблице 5.

Таблица 5. RVN-грамматики для языка ПГС

№	Комплекс-источник	Квази-терм	Комплекс-приемник	RVN-отношение
1	r_0	A_0	r_1	$\emptyset$
2	r_1	rel	r_3	$\emptyset$
3	r_2	$label_P$	r_3	$W_2(b^{1m}, i^{b(4)})$
4		$label_W$	r_3	$W_2(b^{2m}, i^{b(5)})$
5		$label_R$	r_3	$W_2(b^{3m}, i^{b(6)})$
6		$label_L$	r_3	$W_2(b^{4m}, i^{b(7)}, k^{t(3)}) / W_3(m^{t(2)} == k^{t(3)})$
7	r_2	$find$	r_3	$W_2(t^{5m})$
8		no_label	r_k	*
9	r_3	A	r_1	
10		P_a	r_1	$W_1(t^{1m}, i^{t(4)}, 1^{t(8)}) / W_2(e^{t(8)})$
11		P_b	r_2	$W_2(1^{t(8)})$
12		W_a	r_2	$W_1(t^{2m}, 1^{t(8)}) / W_2(e^{t(1)})$
13		W_b	r_2	$W_1(1^{t(1)}, t^{2m}, 1^{t(8)}, i^{t(5)}) / W_2(e^{t(1)}, e^{t(8)})$
14		$\underline{W}$	r_2	$W_1(inc(m^{t(1)})) \& W_2(i^{t(5)}) / W_3(m^{t(1)} < 2) \& W_2(1^{t(8)})$
15		R_a	r_1	$W_1(t^{3m^{(k-1)}}, i^{t(6)}, 1^{t(8)}) / W_2(e^{t(8)})$
16		R_b	r_2	$W_2(1^{t(8)})$
17		L_a	r_2	$W_1(k^{t(3)}, t^{4m}, 1^{t(8)}) / W_2(e^{t(2)})$
18		L_b	r_2	$W_1(1^{t(2)}, k^{t(3)}, t^{4m}, 1^{t(8)}, i^{t(7)}) / W_2(e^{t(2)}, e^{t(8)})$
19		$\underline{L}$	r_2	$W_1(inc(m^{t(2)})) \& W_2(i^{t(7)}) / W_3(m^{t(2)} < k^{t(3)}) \& W_2(1^{t(8)})$
20		A_k	r_2	$\emptyset$
21	r_k			$\emptyset$

Используемые элементы внутренней памяти (в скобках указаны номера магазинов и лент):

- четыре магазина для хранения информации для связей-меток, исходящих из условия (1), объединения взаимоисключающих ветвей (2), распараллеливания (3) и слияния (4);
- один магазин для хранения списка возможных продолжателей (5);
- четыре ленты для хранения информации об уже проанализированных объединениях взаимоисключающих ветвей (1), слияниях (2), продолжателях анализа (8), количестве входов в слияния (3);
- четыре ленты для хранения индексов меток переходов условия (4), объединения взаимоисключающих ветвей (5), распараллеливания (6) и слияния (7).

Операция $*$ выполняется после анализа последнего символа и представляется следующим образом:

$$* = W_2(e^{1m}) \,\&\&\, W_2(e^{2m}) \,\&\&\, W_2(e^{3m}) \,\&\&\, W_2(e^{4m}) \,\&\&\, W_2(2^{\alpha_i(1)}) \,\&\&$$
$$W_3(m^{\alpha_i(2)} > 2) \,\&\&\, W_3(m^{\alpha_i(2)} == k^{\alpha_i(3)})$$
$$W_2(e^{5m}) \,\&\&\, W_3(m^{\alpha_i(8)} == 1)$$

Обобщенный алгоритм контроля диаграммных языков ППР по RV (RVN)-грамматике приведен на рис. 5 и на рис. 6.

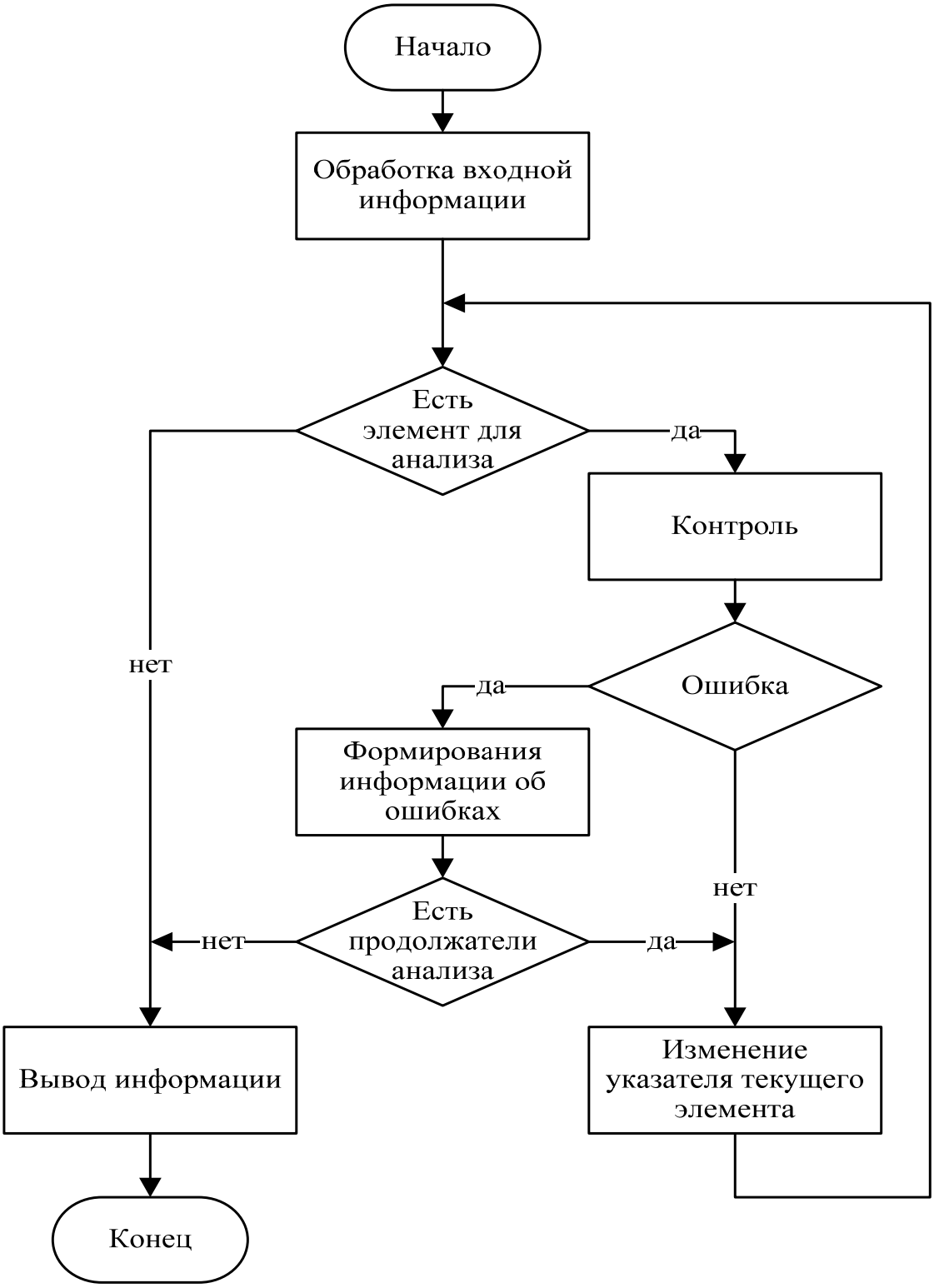


Рис. 5. Алгоритмическая структура RV-анализатора

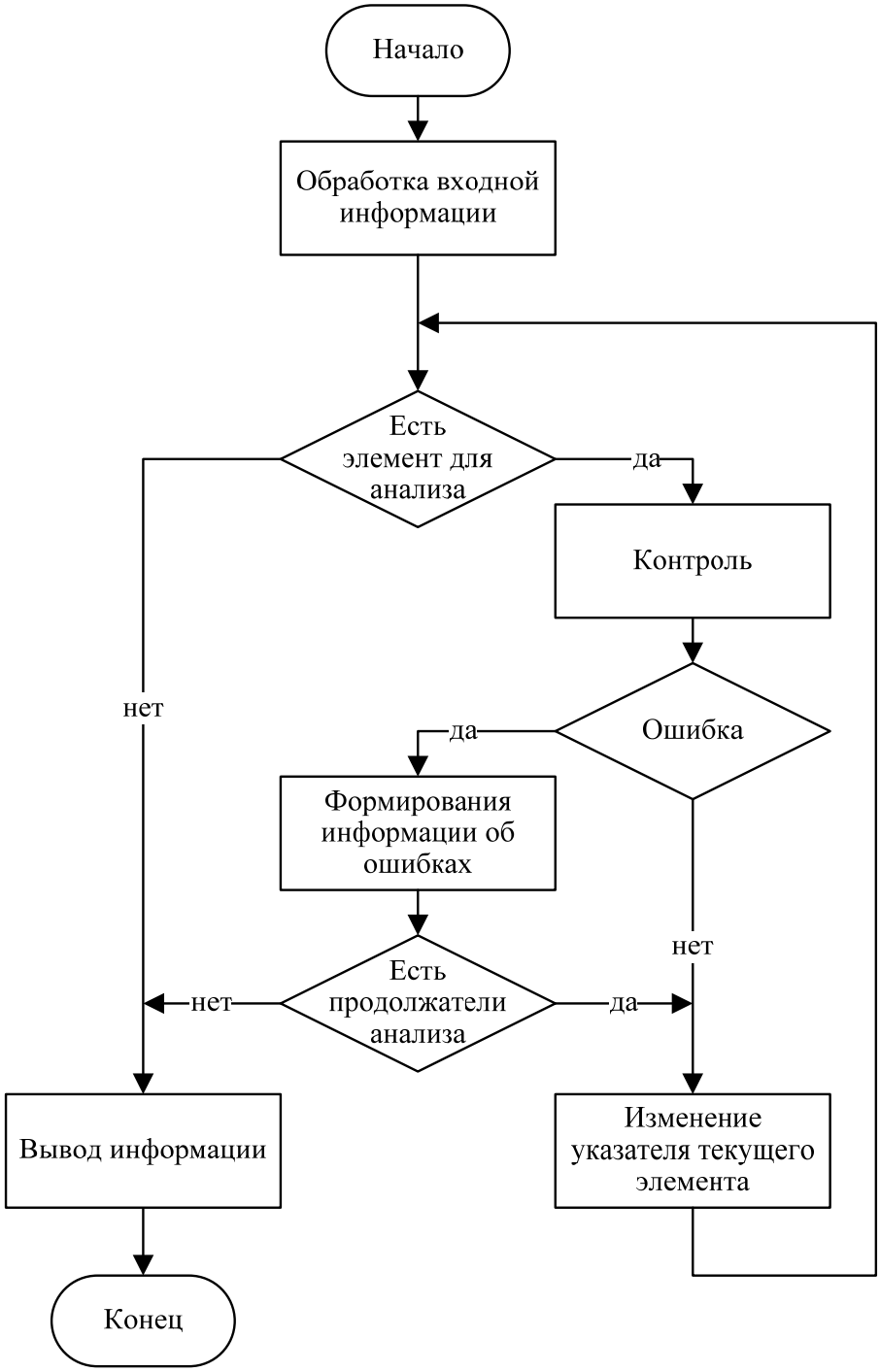


Рис. 6. Алгоритмическая структура RVN-анализатора

Входной информацией RV (RVN)-анализатора являются:

- квазитерминальный алфавит $\tilde{\Sigma}$;
- табличная форма RV (RVN)-грамматики, закодированная определенным образом;

- исходный контролируемый список графических объектов и связей, соединяющих их.

Выходной информацией служат данные:

- о месте синтаксической ошибки (номер ошибочного графического примитива, номер лишней исходящей/входящей связи в графический объект, номер графического объекта, не содержащего необходимой исходящей/входящей связи и т. п.);
- о типе синтаксической ошибки (диагностический текст, поясняющий тип ошибки, например, «графический объект содержит недопустимое количество входящих дуг»);
- об отсутствии ошибок (т. е. анализируемая входная информация принадлежит языку, заданному RV (RVN)-грамматикой).

Основными действиями анализатора являются.

1. Обработка входной информации осуществляет хранение и перекодирование входного формата данных во внутреннее представление.
2. Контроль осуществляет переходы по RV (RVN)-таблице языка и может работать в двух режимах: реверсивный режим используется для меток и поиска продолжателей анализа (сначала выполнение операций над внутренней памятью, затем обработка текущего элемента), нормальный режим – для всех остальных квазитермов (сначала обработка текущего элемента, затем выполнение операций над внутренней памятью).
3. Формирование информации об ошибках осуществляет подготовку сообщения для вывода (указание места и типа ошибки).
4. Вывод информации.

Коллективное проектирование потоков проектных работ связано с разработкой сложных проектов, носящих иерархический характер. В связи с этим ниже предлагаются иерархические RV-грамматики или RVH-грамматики, разработка которых опирается на подобные RG-грамматики.

Построим RVH-грамматику для упрощенного языка граф-схем, в которой не используются циклические конструкции. Такое ограничение позволит реализовать простой и наглядный пример иерархической грамматики.

На рис. 7 представлен набор подграмматик, достаточный для анализа выбранного языка. Двойными кружками обозначены конечные комплексы подграмматик. Дуги графов подграмматик помечены нетерминальными символами, обозначающими синтаксические конструкции, а под дугами показаны соответствующие начальные комплексы подграмматик. Имеются и терминальные символы, помечающие дуги.

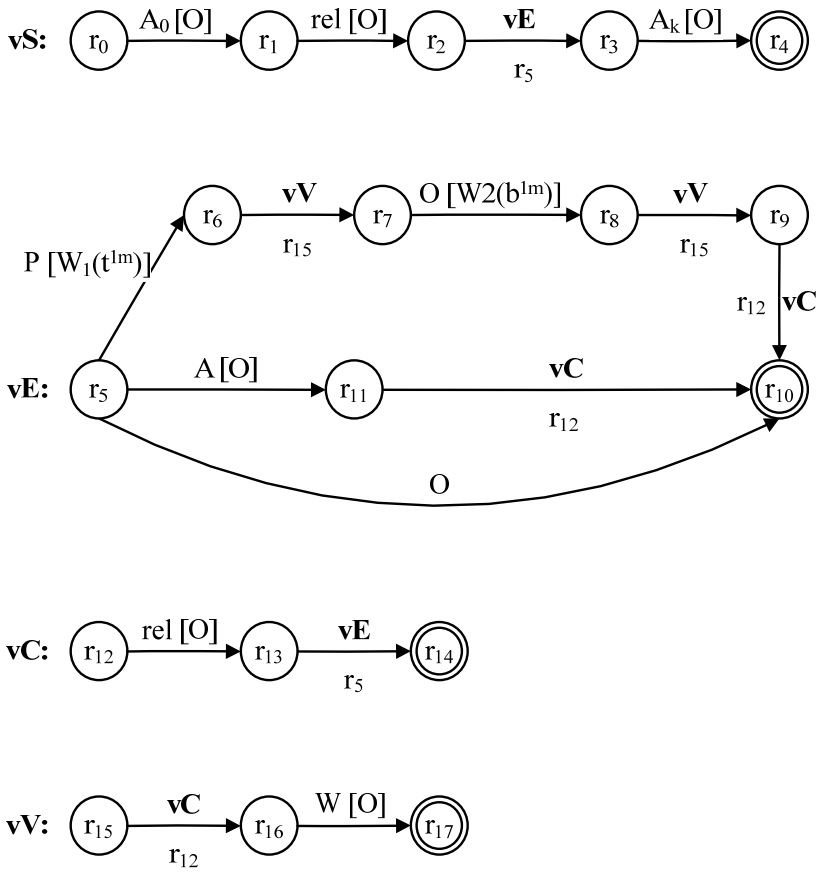


Рис. 7. Иерархическая RVH-грамматика

Последовательность тактов для диаграммы, представленной на рис. 8, имеет вид (в угловых скобках записана конфигурация

RVH-анализатора: текущее состояние анализатора, входной элемент, который способствовал переходу в это состояние и текущее состояние магазина, причем $\$$ – магазинный маркер):

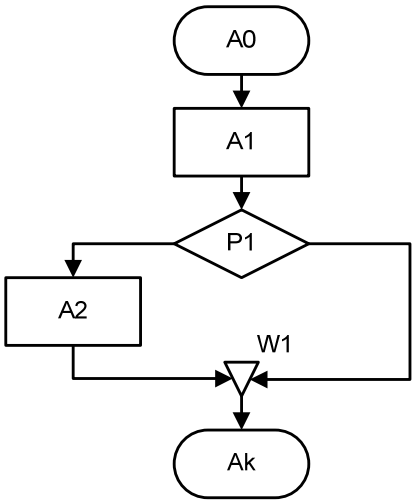


Рис. 8. Пример диаграммы

$$
\begin{aligned}
< r_0, ,\$ > &\mapsto < \eta_1, A_0, \$ > \\
&\mapsto < r_2, rel, \$ > \\
&\overset{C}{\mapsto} < r_5, \downarrow E, r_3\ \$ > \\
&\mapsto < \eta_{11}, A, r_3\ \$ > \\
&\overset{C}{\mapsto} < \eta_{12}, \downarrow C, \eta_{10}\ r_3\ \$ > \\
&\mapsto < \eta_{13}, rel, \eta_{10}\ r_3\ \$ > \\
&\overset{C}{\mapsto} < r_5, \downarrow E, \eta_{14}\ \eta_{10}\ r_3\ \$ > \\
&\mapsto < r_6, P, \eta_{14}\ \eta_{10}\ r_3\ \$ > \\
&\overset{C}{\mapsto} < \eta_{15}, \downarrow V, r_7\ \eta_{14}\ \eta_{10}\ r_3\ \$ > \\
&\overset{C}{\mapsto} < \eta_{12}, \downarrow C, \eta_{16}\ r_7\ \eta_{14}\ \eta_{10}\ r_3\ \$ > \\
&\mapsto < \eta_{13}, rel, \eta_{16}\ r_7\ \eta_{14}\ \eta_{10}\ r_3\ \$ > \\
&\overset{C}{\mapsto} < r_5, \downarrow E, \eta_{14}\ \eta_{16}\ r_7\ \eta_{14}\ \eta_{10}\ r_3\ \$ >
\end{aligned}
$$

$$
\begin{aligned}
&\mapsto && < n_{11}, A, n_{14}\ n_{16}\ r_7\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{C}{\mapsto} && < n_{12}, \downarrow C, n_{10}\ n_{14}\ n_{16}\ r_7\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\mapsto && < n_{13}, rel, n_{10}\ n_{14}\ n_{16}\ r_7\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{C}{\mapsto} && < r_5, \downarrow E, n_{14}\ n_{10}\ n_{14}\ n_{16}\ r_7\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\mapsto && < n_{10}, \oslash, n_{14}\ n_{10}\ n_{14}\ n_{16}\ r_7\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < n_{14}, \ , n_{10}\ n_{14}\ n_{16}\ r_7\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < n_{10}, \ , n_{14}\ n_{16}\ r_7\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < n_{14}, \ , n_{16}\ r_7\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < n_{16}, \ , r_7\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\mapsto && < n_{17}, W, r_7\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < r_7, \ , n_{14}\ n_{10}\ r_3\ \$ > \\
&\mapsto && < r_8, \oslash, n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{C}{\mapsto} && < n_{15}, \downarrow V, r_9\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{C}{\mapsto} && < n_{12}, \downarrow C, n_{16}\ r_9\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\mapsto && < n_{13}, rel, n_{16}\ r_9\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{C}{\mapsto} && < r_5, \downarrow E, n_{14}\ n_{16}\ r_9\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\mapsto && < n_{10}, \oslash, n_{14}\ n_{16}\ r_9\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < n_{14}, \ , n_{16}\ r_9\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < n_{16}, \ , r_9\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\mapsto && < n_{17}, W, r_9\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < r_9, \ , n_{14}\ n_{10}\ r_3\ \$ > \\
&\overset{C}{\mapsto} && < n_{12}, \downarrow C, n_{10}\ n_{14}\ n_{10}\ r_3\ \$ > \\
&\mapsto && < n_{13}, rel, n_{10}\ n_{14}\ n_{10}\ r_3\ \$ >
\end{aligned}
$$

$$
\begin{aligned}
&\overset{C}{\mapsto} && < r_5, \downarrow E, \eta_4\ \eta_0\ \eta_4\ \eta_0\ r_3\ \$ > \\
&\mapsto && < \eta_0, \oslash, \eta_4\ \eta_0\ \eta_4\ \eta_0\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < \eta_4, \ , \eta_0\ \eta_4\ \eta_0\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < \eta_0, \ , \eta_4\ \eta_0\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < \eta_4, \ , \eta_0\ r_3\ \$ > \\
&\overset{R}{\mapsto} && < \eta_0, \ , r_3\ \$ > \\
&\overset{R}{\mapsto} && < r_3, \ , \$ > \\
&\mapsto && < r_4, A_k, \$ >
\end{aligned}
$$

Знаками $\overset{C}{\longmapsto}$ и $\overset{R}{\longmapsto}$ обозначены соответственно продукции обычного типа, вызов (CALL) подграмматики и возврат (RETURN) в подграмматику, начальный комплекс которой находится в вершине магазина. В конце распознавая магазин пуст.

Использование RVH-грамматик позволяет легко модифицировать грамматику, эффективно конструировать грамматику из уже имеющихся подграмматик. Разбиение грамматики на уровни иерархии адекватно отражает иерархическую структуру контролируемой входной информации ППР.

Приведем временные оценки эффективности RV-анализатора.

RV-анализатор обладает линейной временной характеристикой контроля, т. е. время, затрачиваемое на анализ входного списка элементов диаграммы, определяется линейной функцией от его длины и вычисляется по формуле:

$$t = c \cdot L_s \quad \text{или} \quad t = c \cdot k \cdot L_k, \quad k = L_s / L_k, \qquad (1)$$

где c – константа реализации алгоритма, показывающая, сколько команд (операторов) затрачивается на анализ одного объекта при реализации алгоритма контроля на конкретной ПЭВМ.

Количество переходов по состояниям автомата (L_s) и количество элементов диаграммы (L_k) определяются по следующим формулам:

$$L_k = \sum_{i=1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right) \tag{2}$$

$$L_s = \sum_{i=1}^{m}\left(\sum_{j=1}^{V_i} v\_in_{ij} + \sum_{j=1}^{V_i} v\_out_{ij}\right) +$$

$$\sum_{i=m+1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right) + no\_label \tag{3}$$

где V_i – количество графических объектов i -го типа;

v_in_{ij} – количество входов в j -й графический объект i -го типа;

v_out_{ij} – количество выходов из j -го графического объекта i -го типа;

t – общее количество типов объектов;

m – количество типов объектов, имеющих более одного выхода (количество типов меток), используемых для конкретной реализации диаграммы;

$$no\_label = \begin{cases} 1, \text{если } m > 0, \\ 0, \text{если } m = 0. \end{cases}$$

Полагая $k = L_s L_k$,

$$k = \frac{\sum_{i=1}^{m}\left(\sum_{j=1}^{V_i} v\_in_{ij} + \sum_{j=1}^{V_i} v\_out_{ij}\right) + \sum_{i=m+1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right) + no\_label}{\sum_{i=1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right)},$$

получим зависимость общего времени контроля:

$$t = c \cdot k \cdot L_k .$$

Если в исходном списке нет графических объектов, имеющих более одного входа или выхода ($m = 0$), то получим нижнюю оценку времени.

$$k = \frac{\sum_{i=1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right)}{\sum_{i=1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right)} = 1.$$

Значение k достигает максимума в случае, если все типы вершин графического языка имеют более одного входа и выхода ($m = t$).

$$k = \frac{\sum_{i=1}^{t}\left(\sum_{j=1}^{V_i} v\_in_{ij} + \sum_{j=1}^{V_i} v\_out_{ij}\right) + no\_label}{\sum_{i=1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right)}.$$

В этом случае

$$\sum_{j=1}^{V_i} v\_in_{ij} = \sum_{j=1}^{V_i} v\_out_{ij}.$$

Полагая $edges = \sum_{i=1}^{t}\sum_{j=1}^{V_i} v\_out_{ij}$, получим следующую зависимость:

$$k = \frac{no\_label + 2 \cdot edges}{\sum_{i=1}^{t} V_i + edges};$$

$$k = \frac{no\_label}{\sum_{i=1}^{t} V_i + edges} + \frac{2 \cdot edges}{\sum_{i=1}^{t} V_i + edges};$$

$k < 2$, т.к. $\frac{no\_label}{\sum_{i=1}^{t} V_i + edges}$ близко к 0, а $\frac{edges}{\sum_{i=1}^{t} V_i + edges} < 1$.

Таким образом, в зависимости от количества типов графических объектов графического языка и количества входящих/исходящих дуг (не исключено, что разных типов) $k \in left1,2right$.

Был проведен эксперимент, в котором было сгенерировано 500 диаграмм ПГС, затем для каждой из них было определено количество сгенерированных элементов и подсчитано количество состояний анализирующего автомата по формулам 2 и 3. Результат эксперимента представлен в виде точечной диаграммы на рис. 9, он подтверждает линейную зависимость количества применений правил грамматики от количества графических объектов и связей. Кроме этого, было определено матожидание коэффициента, равное $E(k) = 1,31$, и среднее отклонение $\Delta_k = 0,062$.

Поскольку RVN-грамматика является развитием RV-грамматики, справедливо заимствование методики расчета ее временных характеристик.

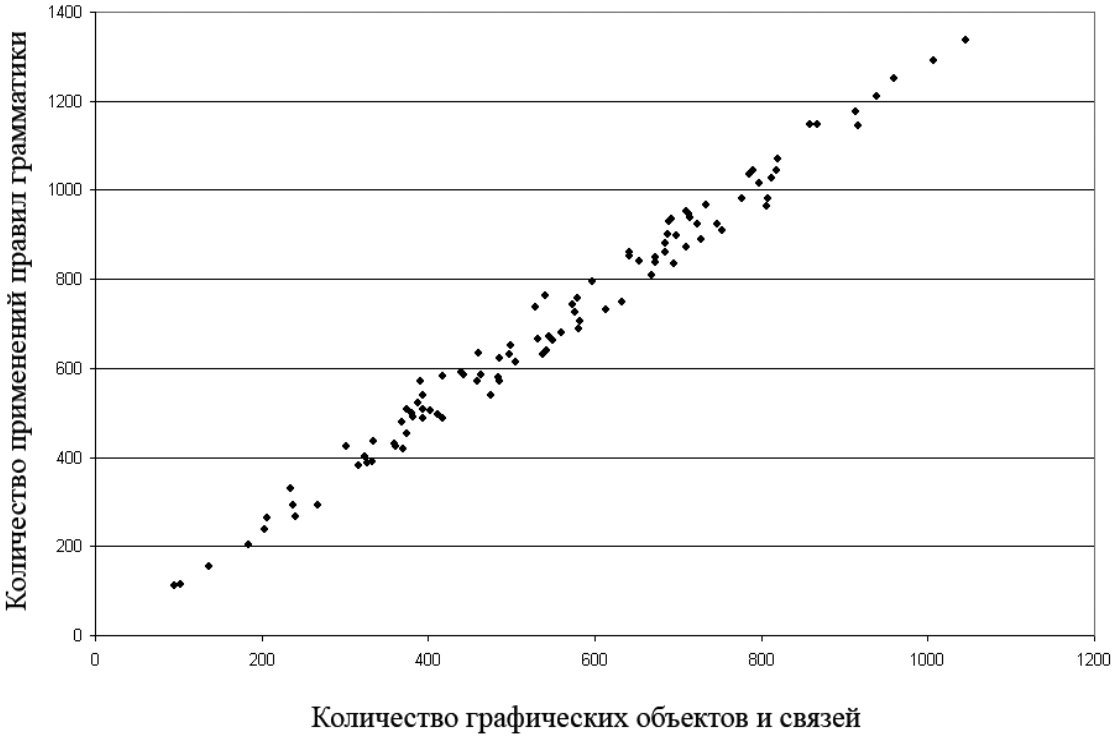


Рис. 9. Значения экспериментальных коэффициентов линейности для языка ПГС

При одном и том же количестве элементов диаграммы (L_k, см. формулу 2) количество переходов по состояниям автомата (L_s, см. формулу 3) для RVN-грамматики будет больше за счет добавления продолжателей анализа, т. е.

$$L_s = \sum_{i=1}^{m}\left(\sum_{j=1}^{V_i} v\_in_{ij} + \sum_{j=1}^{V_i} v\_out_{ij}\right) + \tag{4}$$

$$\sum_{i=m+1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right) + no\_label + c\_errors \tag{5}$$

,

где c_errors зависит от количества ошибок, допущенных при создании диаграммы и изменяется в пределах:

$$c\_errors = \left[0, \left(2 \cdot \sum_{s=1}^{n} V_s\right)\right], \tag{6}$$

где n – количество типов графических объектов-продолжателей анализа;

V_s – количество графических объектов s-го типа;

Используя формулу (1), определяем нижнюю и верхнюю границы временной сложности. Если в исходном списке нет графических объектов, имеющих более одного входа или выхода ($m = 0$) и нет ошибок ($c\_errors = 0$), то получим нижнюю оценку времени.

$$k = \frac{\sum_{i=1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right)}{\sum_{i=1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right)} = 1.$$

Значение k достигает максимума в случае, если все типы вершин графического языка имеют более одного входа и выхода ($m = t$) и допущено максимальное количество ошибок, а, следовательно, используются все продолжатели ($c\_errors = 2 \cdot \sum_{s=1}^{n} V_s$).

$$k = \frac{\sum_{i=1}^{t}\left(\sum_{j=1}^{V_i} v\_in_{ij} + \sum_{j=1}^{V_i} v\_out_{ij}\right) + no\_label + c\_errors}{\sum_{i=1}^{t}\left(V_i + \sum_{j=1}^{V_i} v\_out_{ij}\right)}$$

В этом случае $\sum_{j=1}^{V_i} v\_in_{ij} = \sum_{j=1}^{V_i} v\_out_{ij}$. Полагая $edges = \sum_{i=1}^{t}\sum_{j=1}^{V_i} v\_out_{ij}$, получим следующую зависимость:

$$k = \frac{no\_label + 2 \cdot \sum_{S=1}^{n} V_s + 2 \cdot edges}{\sum_{i=1}^{t} V_i + edges};$$

$$k = \frac{no\_label}{\sum_{i=1}^{t} V_i + edges} + \frac{2 \cdot \sum_{s=1}^{n} V_s}{\sum_{i=1}^{t} V_i + edges} + \frac{2 \cdot edges}{\sum_{i=1}^{t} V_i + edges};$$

где,

$no\_label \sum_{i=1}^{t} V_i + edges$ близко к 0,

$\frac{2 \cdot \sum_{s=1}^{n} V_s}{\sum_{i=1}^{t} V_i + edges} \le 1$ поскольку $n < t$ и $\sum_{i=1}^{t} V_i < edges$,

$\frac{edges}{\sum_{i=1}^{t} V_i + edges} < 1,$

получаем не грубую оценку $k < 3$.

Таким образом, в зависимости от количества типов графических объектов графического языка, количества входящих/исходящих дуг (не исключено, что разных типов) $k \in [1, 2]$, с учетом количества допущенных ошибок это значение изменяется в интервале $[1, 3]$.

Проведем сравнение временных характеристик RV-грамматики с известными формализмами. Позиционная грамматика, предложенная Ченгом и развитая Костаглиолой [80], является контекстно-свободной. Для грамматики определен LR-анализатор, осуществляющий разбор,

чередуя сдвиги и свертки, с помощью стека. Анализатор реализован по аналогии с LR-анализатором текстовых языков, за исключением того, что здесь для перехода к следующему объекту используется алгоритм синтаксически-направленного поиска входа, который был предложен Ченгом для двумерных текстовых языков. В работе анализатор использует внутренние правила для переопределения связей. Временная сложность анализа, по оценкам, предложенным в [81], составляет $O(|G|*(|G|+k))$, где $|G|$ – количество графических объектов в диаграмме, k – максимальное количество проходов анализа через графический объект (соответствует количеству входов графического объекта).

Для реляционной грамматики [82] используется LR-анализатор с экспоненциальной временной сложностью анализа, равной $O(1.5^{|G|})$. Анализатор позволяет обнаруживать более одной ошибки за один проход.

Многоуровневая [83] и сохраняющая [84] графовые грамматики могут быть использованы для порождения или анализа графического языка. За мощность графовых грамматик приходится платить большим временем разбора. Так, для многоуровневой графовой грамматики максимальная временная сложность – $O(2|G|)$, а для сохраняющей графовой грамматики – $O(|G|^{m+1})$, где m – максимальное количество графических объектов в правой части продукции грамматики.

В таблице 6 сведены характеристики рассмотренных грамматик, в том числе и авторской. На рис. 10 показаны графики зависимости временной сложности от количества графических объектов в диаграмме, для рассмотренных грамматик на языке ПГС.

Таблица 6. Сравнительные характеристики грамматик

	Левая часть продукции	**Правая часть продукции**	**Учет контекста**	**Временная сложность**	**Использование сентенциальной формы**	**Нейтрализация ошибок**
Позиционная грамматика	нетерминал	плекс-структура	да	полиномиальная $O((\|G\|(\|G\|+k))$	да	нет
Реляционная грамматика	нетерминал	реляционная структура	да	экспоненциальная $O(1.5^{\|G\|})$	да	да
Многоуровневая графовая грамматика	граф	граф	нет	экспоненциальная $O(2^{\|G\|})$	да	нет
Сохраняющая графовая грамматика	граф	граф	да	полиномиальная $O(\|G\|^{(m+1)})$	да	нет
RV-грамматика	комплекс	комплекс	да	линейная $O(5*\|G\|*2)$	нет	нет
RVN-грамматика	комплекс	комплекс	да	линейная $O(5*\|G\|*3)$	нет	да

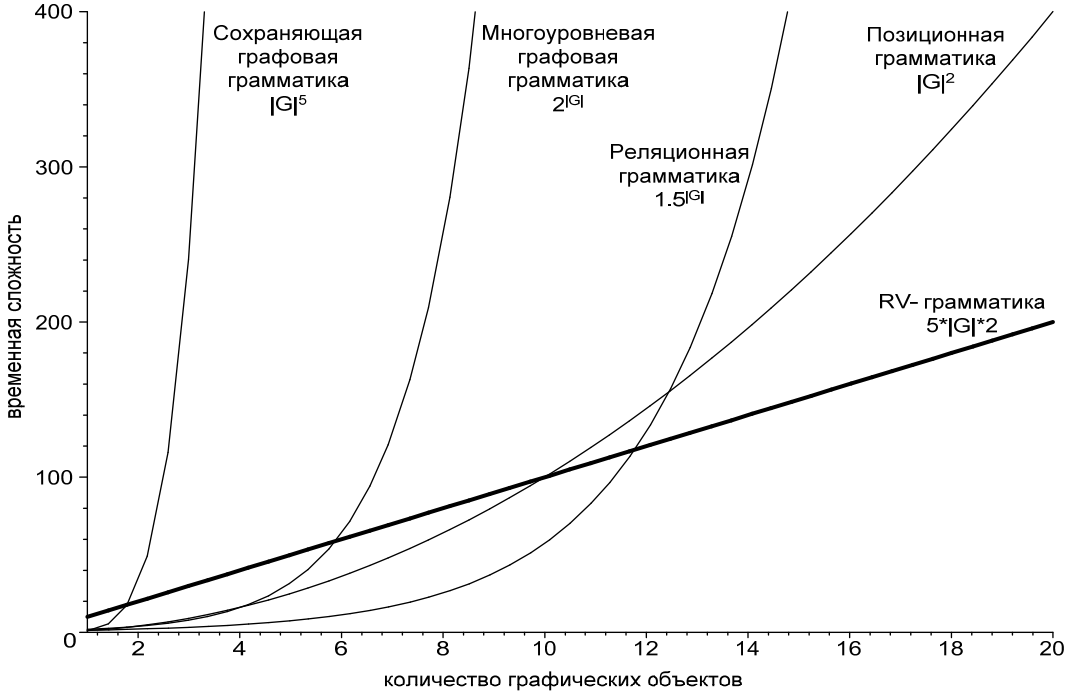


Рис. 10. Сравнение временных характеристик анализаторов

Оценим затраты памяти.

Сравнение затрат памяти анализаторами производится только по объему рабочей памяти анализатора, т. е. по памяти, которая требуется анализатору во время его работы. Расчет затрат памяти на хранение правил грамматики и соответствующих им таблиц разбора не производится, поскольку алгоритм разбора (грамматика) на фазе анализа не хранится в памяти, а является программным кодом. Характеристики анализаторов сведены в таблице 6.

Особенностью RV(x)-анализаторов является то, что в процессе работы анализатор не использует сентенциальную форму, а значит память выделяется только для хранения служебной информации о графических объектах, содержащих более одного входа или выхода.

Объем требуемой памяти рассчитывается по формуле

$$s * \left(\sum_{i=1}^{n} M_i + 2 * \sum_{j=1}^{m} M_j \right),$$

где s – объем ячейки памяти;

$\sum_{i=1}^{n} M_i$ – количество ячеек памяти, хранящих информацию об связях-метках, n – количество типов графических объектов, которым назначаются связи-метки, M_i – количество графических объектов данного типа;

$\sum_{j=1}^{m} M_j$ – количество ячеек памяти, хранящих информацию об уже проанализированных объектах и количестве проход через них.

Формализмы позиционных и реляционных грамматик используют LR-анализатор, которому в работе необходима сентенциальная форма. Для ее хранения необходим объем памяти, достаточный для хранения всех элементов диаграммы. Минимальная структура данных, требуемая для

этого, представлена на рис. 11. Для ее хранения необходим объем памяти, равный

$$L = s * \left(2 * O_c + 2 * C_c + 1\right),$$

где s – объем ячейки памяти; O_c – количество графических объектов диаграммы; C_c – количество связей, используемых в диаграмме.

```
class CDiagram:
{
    CObject    *objs;
};
class CObject:
{
     int    type;
     CConnection     *conns;
};
class CConnection:
{
     CObject     *from;
     CObject     *to;
};
diagram = new CDiagram();
```

Рис. 11. Структура данных для хранения диаграммы

Анализ диаграмм согласно формализмам графовых грамматик осуществляется в два этапа. На первом этапе с помощью восходящего анализа выявляют все возможные LR-выводы, они сохраняются в памяти и используются на втором этапе – этапе нисходящего анализа, который извлекает подмножества выводов для порождения исходного графа. Таким образом, анализатор графовых грамматик использует несколько сентенциальных форм, их количество зависит от сложности сочетаний графических конструкций в диаграмме, и хранит LR-выводы, количество

которых соответствует количеству сентенциальных форм. Результат сравнений представлен в табл. 7.

Таблица 7. Сравнение затрат памяти анализаторами

	Тип анализатора	**Таблицы разбора**	**Внутренняя память**	**Сентен-циальная форма**
Позиционная грамматика	LR	LR-таблица	нет	L
Реляционная грамматика	LR	LR-таблица	нет	L
Многоуровневая графовая грамматика	LR + LL	LR- и LL-таблицы	$k*V, k>1$	$k*L, k>1$
Сохраняющая графовая грамматика	LR + LL	LR- и LL-таблицы	$k*V, k>1$	$k*L, k>1$
RV-грамматика	автомат	V-таблица	$s*\left(\sum_{i=1}^{n} M_i + 2*\sum_{j=1}^{m} M_j\right)$	нет

Был проведен эксперимент, в котором было сгенерировано 500 диаграмм ПГС, затем для каждой из них был определен объем требуемой памяти при анализе RV-анализатором и LR-анализатором, реализованным на базе позиционных/реляционных грамматик. Расчет для графовых анализаторов не проводился, поскольку объем требуемой памяти явно выше чем для LR-анализатора. Результат эксперимента представлен в виде точечной диаграммы на рис. 12.

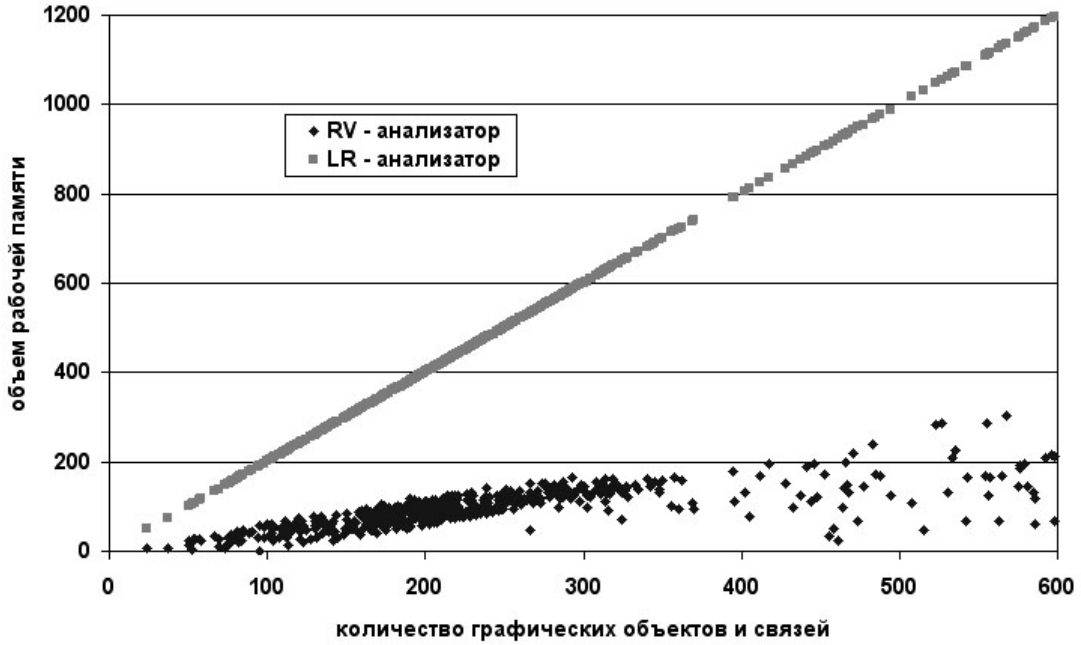
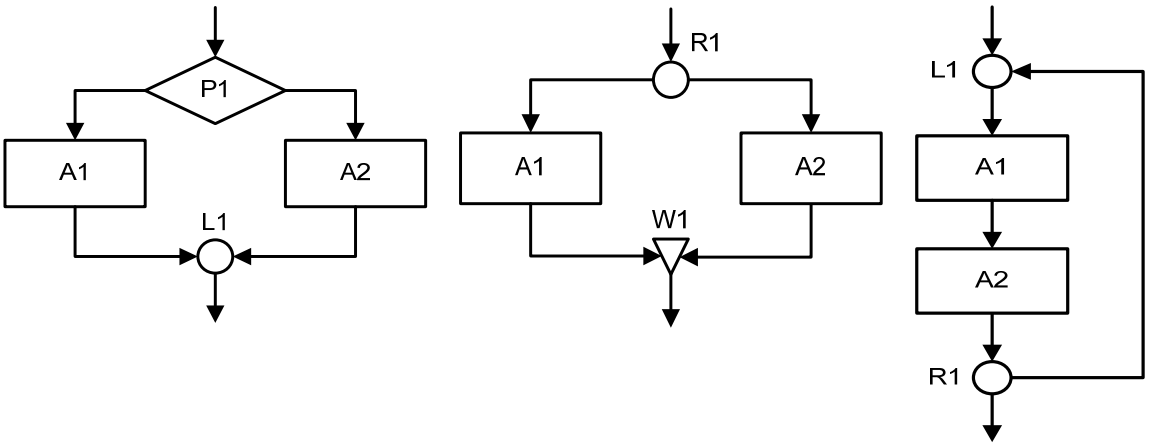


Рис. 12. Зависимость объема памяти от количества элементов диаграммы для RV- и LR-анализаторов

Как отмечалось ранее, использование внутренней памяти позволяет RV(x)-анализаторам выявлять ошибки, связанные с нарушением связности графических объектов, указывать возможные типы продолжателей «оборванной» цепи. В таблице 8 приведены возможные типы таких синтаксических ошибок. Сочетания графических объектов в них может быть разным, оно определяется грамматикой и определяет конечное сообщение об ошибке. На рисунках знаком вопроса помечается отсутствие объекта или связи, двумя косыми или скрещенными линиями недопустимость графического объекта или связи.

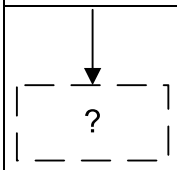
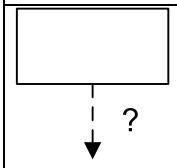
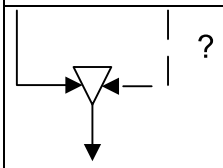
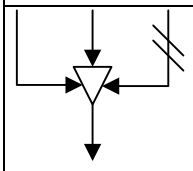
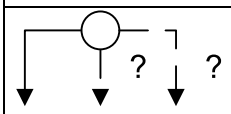
Кроме этого, определяются семантические ошибки: зависание, неоднозначность, дедлок. Их графическое представление в терминах языка ПГС показано на рис. 13.



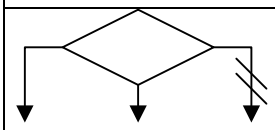
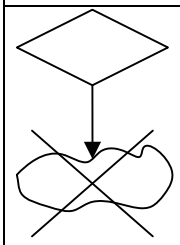
а б в

Рис. 13. Семантические ошибки ПГС:
зависание (а), неоднозначность (б), дедлок (в)

Таблица 8. Типы выявляемых синтаксических ошибок

Ошибка	**Описание**
	Ожидался графический объект
	Ожидалась исходящая связь
	Ожидалась входящая связь в графический объект с фиксированным количеством входящих связей
	Обнаружена лишняя входящая связь в графический объект с фиксированным количеством входящих связей
	Недостаточно малое количество исходящих связей для графического объекта с фиксированным количеством исходящих связей или их фиксированным минимальным количеством

Окончание табл. 8

Ошибка	Описание
	Обнаружена лишняя исходящая связь в графический объект с фиксированным количеством исходящих связей
	Ожидался иной графический объект, наличие текущего графического объекта не допустимо

Для анализа растровых представлений диаграмм ППР (например, полученных сканированием технической документации) могут быть использованы предлагаемые нечеткие RV-грамматики (RVF-грамматики) [85], основанные на элементах теории нечетких множеств [86, 87]. В RVF-грамматике комплекс $\tilde{r}_i$ представляет собой нечеткое множество правил $\tilde{p}_{ij}$ и степеней их принадлежности $\mu(\tilde{p}_{ij})$: $\tilde{r}_i = \{(\tilde{p}_{ij}, \mu(\tilde{p}_{ij})), j = \overline{1,J})\}$, где $\mu(\tilde{p}_{ij}) \in [0,1]$. Степень принадлежности цепочки Ψ определяется следующим образом. Пусть цепочка $\Psi = \alpha_{t_1}, \alpha_{t_2}, ..., \alpha_{t_\lambda}$ является *RVF*-выводом α_{t_λ} из α_{t_1}. В каждой подцепочке вывода $\alpha_{t_i}\alpha_{t_{i+1}}$ выбирается минимальное значение степени принадлежности μ_i, а затем из величин μ_i выбирается максимальная по всем выводам. Рассматривается понятие «прочность», понимаемое как $p^\lambda = \min\limits_{\lambda}\{\mu_1, \mu_2, \ldots, \mu_\lambda\}$. Под «прочностью связи» понимается максимальное значение «прочности» среди всех цепочек вывода $\rho = \max\limits_{l}\min\limits_{\lambda}\{\mu_1^l, \mu_2^l, \ldots, \mu_\lambda^l\}$. Величина ρ принимается за степень принадлежности $\mu_G(\Psi)$ цепочки Ψ языку *L(G),* порождаемому нечеткой *RVF*-грамматикой.

3. Разработка методов семантического анализа и трансляции графических языков потоков проектных работ

Большинство синтаксически формальных графических языков не являются семантически формальными. Такие языки гибки и позволяют строить диаграммы, которые могут быть применены в различных предметных областях. Гибкость языкам придает неполнота или неформальность их описания, и, как следствие, результирующие диаграммы можно интерпретировать неоднозначно. Машинная обработка графических диаграмм таких языков затруднена. Гибкость языка может привести к возникновению семейства языков, т. е. множества языков, которые концептуально имеют общую базу, но различную интерпретацию, специфичную для предметной области их применения. Большинство существующих подходов рассматривают такие языки изолированно, хотя достаточно определить обобщающую семантику для семейства языков (возможно для некоторых элементов она будет абстрактной) и специализировать семантическую составляющую отдельных элементов языка и (или) диаграммы перед ее интерпретацией.

На рис. 14 приведена качественная оценка уровня формальности и специализации диаграммных языков.

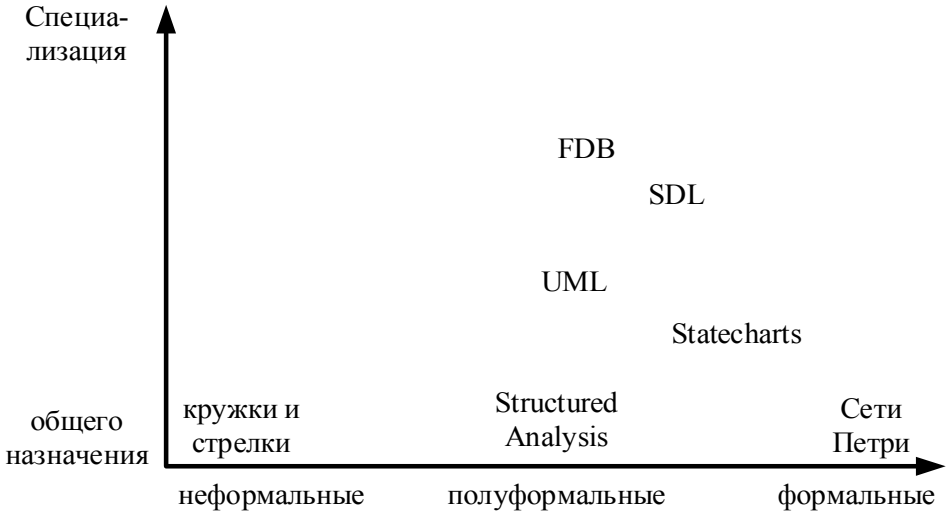


Рис. 14. Формальность против специализации

По абсциссе измеряется формальность языка и выделено три основных типа (статуса) языков.

1. Формальные. Синтаксис и семантики таких языков формально определены.

2. Полуформальные. Синтаксис языка формален, а семантика может иметь разные интерпретации.

3. Неформальные. Синтаксис и семантика языка неформальны.

По ординате задается назначение языка и различают две крайности.

1. Графические языки общего назначения. Обычно применяются для спецификации проектных решений различных предметных областей.

2. Специализированные графические языки. Используются только для специфичных предметных областей.

Из рис. 14 видно, что большинство графических языков ППР являются полуформальными. А значит, на них следует сконцентрировать внимание и проводить исследования по поиску методов их формализации для автоматизации работы с ними. Следует отметить, что это направление активно развивается последние несколько лет у западных исследователей, которые в большинстве своем предлагают придавать семантическую формальность языкам следующими методами.

1. Специализировать язык. Отказаться он некоторых возможностей языка, упростить его, наделить новыми возможностями и т. п., что положительно повлияет на формализацию.

2. Определять семантику динамически. Динамическая семантика предполагает трансформацию диаграмм базового графического языка в некий целевой язык.

В методах предлагаются противоположные подходы, но они являются не взаимоисключающими, а, скорее, взаимодополняющими, т. е. не исключается их совместное использование.

В первом случае – это сужение исходного определения языка, приведение его к состоянию, когда неоднозначность исчезает. Метод

эффективен, когда необходимо реализовать узкоспециализированное, малобюджетное инструментальное средство для автоматизации процесса проектирования, программирования и т. п.

Во втором случае предлагается расшифровать эту неоднозначность и не изменять язык. Этот метод более перспективен, поскольку дает возможность на базе одного универсального инструментального средства реализовывать диаграммы различных графических языков. Метод развивает идеи библиотек примитивов. Однако в данном случае в библиотеках хранятся не новые графические образы, а их интерпретация (отображение) в терминах целевого графического либо текстового языка. Причем может быть несколько интерпретаций для одного и того же графического образа, каждой из них назначается уникальное имя во избежание неоднозначного толкования и некорректного использования. В качестве целевого языка выбирают более формальный язык относительно базового.

Очевидно, что для любого из методов семантической формализации ставится цель – создание достаточных условий для получения семантического значения диаграммы графического языка ППР. В свою очередь, конечный формат представления семантического значения и метод его получения определяются типом семантики.

Для графических языков выделяют следующие типы семантик.

- Денотационная семантика. Трансформация графической диаграммы в семантические домены (денотаты) есть процесс получения денотационной семантики. Денотаты рассматриваются как идеализированные математические объекты, моделирующие (представляющие) другие объекты. Для графических языков в качестве семантических доменов (денотатов) выбирают формальные языки: сети Петри, CSP (Communicating Sequential Processes).

• Операционная семантика. Смысл графической диаграммы задается посредством описания процессов ее исполнения. Определение операционной семантики подразумевает получение из графической диаграммы отображения в терминах некоторой абстрактной вычислительной машины и правил выполнения этого отображения на ней. Отображением может являться исходный код на каком-либо языке программирования, набор математических формул, специализированная алгебра и т. п.

• Аксиоматическая семантика. Сосредотачивает внимание на выборе и исследовании постулатов (аксиом и правил вывода), которые обеспечивают универсум для рассуждения о программах.

• Алгебраическая семантика. Рассматривается как ветвь денотационной семантики, привлекающая существенно алгебраические понятия и методы.

• Лингвистическая семантика. Денотативная и сигнификативная семантики содержатся в лингвистической семантике.

В рамках данного исследования рассматриваются денотационная, операционная, денотативная и сигнификативная семантики. Они предназначены для приведения исходного описания путем трансляции к более формальному описанию для последующего анализа семантической корректности диаграммы ППР.

RVT-грамматика – это транслирующая грамматика, позволяющая производить синтаксически-ориентированную трансляцию диаграмм графического языка в текстовые и графические формальные описания. Для повышения эффективности реализации инструментальных средств на базе RVT-грамматики вводится два типа RVT-грамматик.

1. RVTt-грамматика. Грамматика, для которой целевым является текстовое формальное описание (программы на языке программирования, текст на специфичном языке разметки и т. п.).

2. RVTg-грамматика. Грамматика, позволяющая формировать выходные цепочки в терминах графических языков.

3. RVT-грамматика является развитием RV-грамматики, в котором продукции схемы грамматики расширены для хранения в них соответствия в терминах целевого формального описания, а внутренняя память хранит информацию необходимую для процесса трансляции.

Определение 3. RVTt-грамматикой [87] языка L(G) называется упорядоченная семерка непустых множеств $G = (V, U, \Sigma, \tilde{\Sigma}, M, R, r_0)$, где

- $V = \{v_e, e = \overline{1, L}\}$ – алфавит операций над внутренней памятью базового языка;
- $U = \{v_e, e = \overline{1, K}\}$ – алфавит операций над внутренней памятью, используемый целевым языков;
- $\Sigma = \{a_t, t = \overline{1, T}\}$ – терминальный алфавит графического языка, являющийся объединением множеств его графических объектов и связей (множество примитивов графического языка);
- $\tilde{\Sigma} = \{\tilde{a}_t, t = \overline{1, \tilde{T}}\}$ – квазитерминальный алфавит, являющийся расширением терминального алфавита. Алфавит $\tilde{\Sigma}$ включает:

– квазитермы графических объектов;

– квазитермы графических объектов, имеющих более одного входа;

– квазитермы связей-меток с определенными для них семантическими различиями.

– квазитерм, определяющий отсутствие связей-меток.

- $M = TT \cup TN$ – объединение алфавитов терминальных (TT) и нетерминальных (TN) символов целевого языка.
- $R = \{r_i, i = \overline{0, I}\}$ – схема грамматики G (множество имен комплексов продукций, причем каждый комплекс r_i состоит из подмножества P_{ij} продукций $r_i = \{P_{ij}, j = \overline{1, J}\}$);

• $r_0 \in R$ – аксиома RVTt-грамматики (имя начального комплекса продукций), $r_k \in R$ – заключительный комплекс продукций.

Множества $V, \Sigma, \tilde{\Sigma}, R, r_0$ наследуются от RV-грамматики и используются для определения синтаксический корректности анализируемой диаграммы.

Множества U, M необходимы для придания грамматике транслирующих функций. Именно они и создают новый формализм, расширяя RV-грамматику.

Продукция $P_{ij} \in r_i$ имеет вид

$$P_{ij} : \tilde{a}_t \xrightarrow{\Omega_\mu\left[W_\nu(\gamma_1,..,\gamma_n)\right]\{\Theta_\mu\left[W_\nu(\gamma_1,..,\gamma_n)\right]\}} r_m\{\chi\},$$

где $W_\nu(\gamma_1,...,\gamma_n)$ – n-арное отношение, определяющее вид операции над внутренней памятью в зависимости от $\nu \in \{0,1,2,3\}$;

Ω_μ (Θ_μ) – оператор модификации, определенным образом изменяющий вид операции над памятью базового (целевого) языка, причем $\mu \in \{0,1,2\}$;

$r_m \in R$ – имя комплекса продукции-преемника;

χ – отображение квазитерма в терминах целевого языка (набор символов $m \in M$).

Методика построения RV-грамматики для RVTt-грамматики расширяется на фазе синтеза следующими этапами:

• определение отображения квазитерма в терминах целевого языка для каждой продукции, т. к. один и тот же квазитерм в зависимости от контекста может интерпретироваться разным набором символов целевого языка;

• определение операций над внутренней памятью для элементов целевого языка.

На этапе анализа устранение недетерминированности и неопределенности производится по аналогии с соответствующими операциями RV-грамматики. А вот задача минимизации грамматики претерпевает некоторые изменения и для RVT-грамматики осуществляется как циклическое применение двух типов минимизаций:

- минимизация количества состояний;
- минимизация отображений.

Этап минимизации количества состояний соответствует этапу минимизации RV-грамматики. Однако в данном случае эквивалентными состояниями считаются состояния, у которых, кроме одинаковых продолжений распознаваемой цепочки справа и операций над внутренней памятью, одинаковыми являются транслирующие операции над внутренней памятью и отображения в терминах целевого языка. Сама процедура минимизации остается неизменной.

Минимизация отображений предполагает поиск и свертку эквивалентных цепочек символов терминального и нетерминального алфавитов целевого языка. Более развернуто эту процедуру можно описать как последовательность следующих действий:

- выделение эквивалентных фрагментов (цепочек или подцепочек) отображений и назначение каждому классу эквивалентности нового нетерминального символа;
- замена эквивалентных фрагментов соответствующими нетерминальными символами.

В качестве возможных эквивалентных фрагментов могут использоваться цепочки, не содержащие символов, для которых необходимо применять операции над внутренней памятью.

Специфика трансляции в текстовый целевой язык состоит в том, что символы результирующей цепочки должны быть выстроены в определенной последовательности.

Большинство графических языков «выросли» из текстовых языков, преобразовав текстовые конструкции в графические, поэтому трансляция графического языка в его текстовый эквивалент очень проста. При этом результирующее текстовое описание может быть избыточным относительно аналогичного, но реализованного техническим специалистом. Эти издержки не являются признаком некачественной трансляции, это обычная практика при преобразовании кода/описания на языке высокого уровня в код/описание на языке низкого уровня, в данном случае графического и текстового языков, соответственно. Если есть потребность в устранении избыточности, то необходимо использовать подпрограммы минимизации, адаптированные для этого конкретного целевого языка.

Сложности преобразования появляются в случае, когда базовый и целевой язык являются разнородными. Здесь, если не получается средствами грамматики определить требуемое соответствие, то для решения проблемы необходимо применить один или несколько следующих методов:

- приведение конструкций графического языка к конструкциям целевого текстового языка;
- использование подпрограмм дефрагментации выходной цепочки и функций постобработки;
- использование промежуточного языка.

Первый метод предполагает приведение базового языка к новому его варианту, такому, чтобы его конструкции находили однозначное отражение в терминах целевого языка. Этот метод очень простой, но не всегда эффективный, поскольку базовый язык, как правило, уже является сформировавшимся, нашедшим свое применение на практике. Введение новых конструкций потребует внесение изменений в спецификацию языка, обучение персонала и т. п.

Второй метод очень сложен в практической реализации. Поскольку кроме того, что такие подпрограммы уникальны и применимы только для

конкретного целевого языка, они еще предполагают привлечение программистов высокого уровня, их реализация не тривиальна.

Следует отметить, что первые два метода по реализации много затратнее третьего, который предполагает использование промежуточных графических, либо текстовых формальных описаний. Этот метод хорош тем, что позволяет использовать уже накопленную базу инструментальных средств, но процедура трансляции в этом случае в конечный формат возможно будет затратнее по времени.

Далее исследуются отличия в построении RVTt- и RV-грамматик. Описание терминального и квазитерминального алфавита представлено в таблице 1, табличная форма RV-грамматики – в таблице 3. Последняя расширяется до RVTt-грамматики введением отображений в терминах языка ПГС и операций над внутренней памятью для целевого языка.

Из табличной формы грамматики видно, что каждому квазитерму соответствует одно или более правил грамматики. А значит, одному и тому же квазитерму может быть поставлено в соответствие одно или несколько отображений, состоящих из элемента или конструкции элементов алфавита целевого языка.

Формируя алфавит внутренней памяти целевого языка, следует выделить четыре эластичные ленты для хранения информации об индексах графи-ческих объектов, которыми впоследствии наделяются исходящие связи из логического условия (лента 4) и распараллеливания (6), входящие и исхо-дящие связи из объединения взаимоисключающих ветвей (5) и слияния (7), и ленту (9) из трех ячеек для хранения следующего индекса соответствующего типа. Ячейка 1 хранит индекс для связей-меток исходящих из логических условий и объединения взаимоисключающих ветвей, ячейка 2 – распаралле-ливаний, ячейка 3 – слияний. Перед началом анализа эти ячейки заполняются символом «1».

В данном случае операции над внутренней памятью имеют некоторую общность и их можно сгруппировать следующим образом.

1. Операции назначения индекса графическим объектам

(логическому условию, объединению взаимоисключающих ветвей, распараллеливанию и слиянию). $W_2(i^{2(9)}) \& W_1(i^{t(6)}, inc(i^{2(9)}))$ – чтение следующего индекса для связи-метки из ленты 9, запись этого индекса в соответствующую объекту эластичную ленту, увеличение значения следующего индекса на единицу.

2. Операции определения индекса при повторном анализе графических объектов, содержащих более одного входа (объединение взаимоисключающих ветвей и слияние). $W_2(i^{t(7)})$ – чтение индекса анализируемого объекта, для указания принадлежности входящих в него связей.

3. Операции определения индекса исходящей из графического объекта связи-метке. $W_2(i^{b(4)})$ – чтение индекса для связей-меток при изъятии таковой из магазина в качестве продолжателя анализа.

После анализа последнего символа входной цепочки выполняются операции $*$ и $**$. Первая, проверяя состояние памяти, принадлежащей алфавиту V, обеспечивает обнаружение синтаксических ошибок. Вторая, проверяя состояние памяти алфавита U, выявляет ошибки, произошедшие на этапе трансляции.

$$* = W_2(e^{1m}) \& \& W_2(e^{2m}) \& \& W_2(e^{3m}) \& \& W_2(e^{4m}) \& \& W_2(2^{\alpha_i(1)}) \& \&$$
$$W_3(m^{\alpha_i(2)} > 2) \& \& W_3(m^{\alpha_i(2)} == k^{\alpha_i(3)})$$
$$* = \oslash$$

Операция $**$ в данном случае пустая, так как в памяти хранятся только индексные значения, проверка которых не может дать ни отрицательного, ни положительного ответа на результаты трансляции.

Расширив исходную табличной формы RV-грамматики (см. табл. 2) двумя столбцами справа (операции над внутренней памятью целевого языка и отображения квазитермов в терминах целевого языка), произведя минимизацию согласно описания RVTt-грамматики, получаем представленную в таблице 9 табличную форму транслирующей

RVTt-грамматики для языков ПГС и параллельных логических схем (ПЛС).

Таблица 9. RVTt-грамматика для трансляции ПГС в ПЛС

№	Комп-лекс	Квази-терм	Прием-ник	RVTt-отношения		Отобра-жение
				базовый язык	целевой язык	
1	r_0	A_0	r_1	$\emptyset$		A_0
2	r_1	rel	r_3	$\emptyset$		
3	r_2	$label_P$	r_3	$W_2(b^{1m})$	$W_2(i^{b(4)})$	$)i$
4		$label_W$	r_3	$W_2(b^{2m})$	$W_2(i^{b(5)})$	$)i$
5		$label_R$	r_3	$W_2(b^{3m})$	$W_2(i^{b(6)})$	$]i$
6		$label_L$	r_3	$W_2(b^{4m}) / W_3(m^{t(2)} = k^{t(3)})$	$W_2(i^{b(7)}, k^{t(3)})$	$\}i / k\ L$
7		no_label	r_k	*	* *	
8	r_3	A	r_1	$\oslash$		A
9		P	r_1	$W_1(t^{1m})$	$W_2(i^{1(9)})$ & $W_1(i^{t(4)}, inc(i^{1(9)}))$	$P(i$
10		W	r_2	$W_1(1^{t(1)}, t^{2m}) / W_2(e^{t(1)})$	$W_2(i^{1(9)})$ & $W_1(i^{t(5)}, inc(i^{1(9)}))$	$w(i$
11		$\breve{W}$	r_2	$W_1(2^{t(1)}) / W_2(1^{t(1)})$	$W_2(i^{t(5)})$	$w(i$
12		R	r_1	$W_1(t^{3m^{(k-1)}}) / W_3(k > 1)$	$W_2(i^{2(9)})$ & $W_1(i^{t(6)}, inc(i^{2(9)}))$	$R\ w[i / k\ ]i$
13		L	r_2	$W_1(1^{t(2)}, k^{t(3)}, t^{4m})$ $/W_2(e^{t(2)})$	$W_2(i^{3(9)})$ & $W_1(i^{t(7)}, inc(i^{3(9)}))$	$w\{i$
14		$\breve{L}$	r_2	$W_1(inc(m^{t(2)}))$ / $W_3(m^{t(2)} < k^{t(3)})$	$W_2(i^{t(7)})$	$w\{i$
15		A_k	r_2	$\emptyset$		A_k
16	r_k			$\emptyset$		

Определение 4. RVTg-грамматика.

RVTg-грамматикой [87] языка L(G) называется упорядоченная восьмерка непустых множеств $G = (V, U, \Sigma, \tilde{\Sigma}, M, F, R, r_0)$, где

- $V = \{v_e, e = \overline{1, L}\}$ – алфавит операций над внутренней памятью базового языка;

- $U = \{\nu_e, e = \overline{1, K}\}$ – алфавит операций над внутренней памятью, используемый целевым языком;
- $\Sigma = \{a_t, t = \overline{1, T}\}$ – терминальный алфавит графического языка, являющийся объединением множеств его графических объектов и связей (множество примитивов графического языка);
- $\tilde{\Sigma} = \{\tilde{a}_t, t = \overline{1, \tilde{T}}\}$ – квазитерминальный алфавит, являющийся расширением терминального алфавита. Алфавит $\tilde{\Sigma}$ включает:
  - квазитермы графических объектов;
  - квазитермы графических объектов, имеющих более одного входа;
  - квазитермы связей-меток с определенными для них семантическими различиями.
  - квазитерм, определяющий отсутствие связей-меток.
- $M = TT \cup TN$ – объединение алфавитов терминальных (TT) и нетерминальных (TN) символов целевого языка.
- $F = \{generate\_input(), generate\_output(), select\_output(), stick\_connection\_points()\}$ – множество транслирующих функций по работе с элементами множества M
- $R = \{r_i, i = \overline{0, I}\}$ – схема грамматики G (множество имен комплексов продукций, причем каждый комплекс r_i состоит из подмножества P_{ij} продукций $r_i = \{P_{ij}, j = \overline{1, J}\}$);
- $r_0 \in R$ – аксиома RVTg-грамматики (имя начального комплекса продукций), $r_k \in R$ – заключительный комплекс продукций.

RVTg-грамматику можно назвать расширением RVTt-грамматики для графических языков. Она наследует правила формирования продукций грамматики, а также все этапы фаз синтеза и анализа грамматики.

Отличием же является то, что элементы множества M – это графические примитивы, меняется смысл хранимых данных во внутренней памяти (U) для целевого языка, а также добавляется множество транслирующих функций.

В ячейках внутренней памяти алфавита U хранятся либо идентификаторы точек соединения графических объектов, из которых исходит связь-метка или которые ожидают еще одну или более входящую связь, либо идентификатор самого графического объекта. На этапе реализации анализатора этот выбор должен быть сделан разработчиком. Детализация до точек соединения может быть полезна, когда необходимо привязаться к геометрии графического объекта целевого языка.

Графические объекты, содержащие более одного входа или выхода, нагружаются транслирующими функциями из множества F, для обеспечения соответствия входов и выходов таких объектов базового и целевого языков.

Назначение транслирующих функций и выполняемые ими операции.

1. *generate_input*() – формирование набора входных точек соединения, кроме той, по которой был достигнут данный графический объект. Выполняется при первичном анализе графических объектов, содержащих более одного входа.

2. *generate_output*() – формирование набора исходящих точек соединения. Выполняется при первичном анализе графических объектов, содержащих более одного выхода, либо когда единственный выход предполагается использовать как связь-метку, т. е. необходимо изменить направление анализа.

3. *select_output*() – выбор исходящей точки соединения элемента в качестве продолжателя цепочки целевого языка из точек соединения для исходящих связей. Событийная функция, выполняемая для графических объектов с динамически изменяемым числом исходящих точек

соединения, используется после формирования набора точек соединения для исходящих связей. В общем случае алгоритм выбора не регламентируется, т. е. выбор является случайным.

4. *stick_connection_points*() – связывание точек соединения связи и объекта. Событийная функция, выполняемая при вторичном анализе графических объектов, содержащих более одного входа. Производит связывание входящей связи с точкой соединения объекта, информация о которой хранится во внутренней памяти.

Наличие этих функций позволяет сформировать алгоритм построения выходной цепочки, основными операциями в котором являются выбор точки-продолжателя анализа для объектов, содержащих более одного выхода, и компоновка целостной последовательности из уже проанализированных объектов, содержащих более одного входа, и связываемых с ними анализируемых объектов. Порядок применения функций представлен на рис. 15.

Особого внимания заслуживает процедура выбора точки продолжателя построения целостной выходной графической диаграммы (см. рис. 15). Понятно, что эта процедура второстепенна относительно анализа входной цепочки и напрямую зависит от нее, но для выходной цепочки она имеет определяющее значение.

Каждое отображение, поставленное в соответствие правилу грамматики, может содержать одну исходящую точку, которая явно указывает направление формирования выходной цепочки; назовем ее направляющей точкой. Такой точки может и не быть, например, если анализ данного графического объекта еще не завершен, не проанализированы все исходящие связи, т. е. для продолжения анализа предполагается выбрать одну из связей-меток, собранных ранее, или графический объект содержит несколько исходящих точек и их

количество, не фиксированное для данного типа графических объектов; в этом случае для выбора такой точки может быть задействована транслирующая функция *select_output*(). Если объект имеет исходящие точки, кроме направляющей, эти точки будут обработаны функцией *generate_output*() и использованы в качестве связей-меток, когда не будет явных продолжателей анализа.

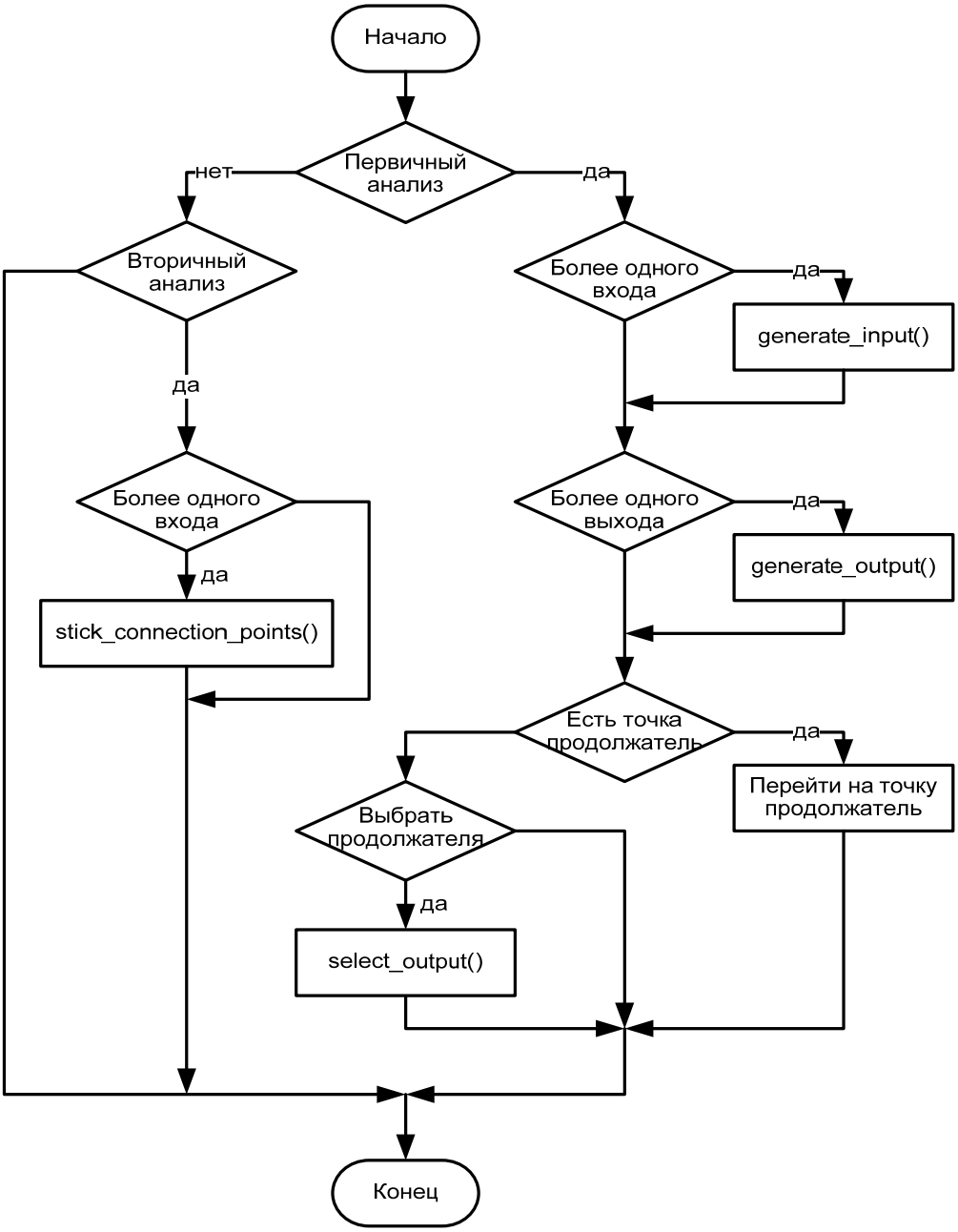


Рис. 15. Порядок применения транслирующих функций
при анализе графического объекта

Входящие точки, отличные от той, по которой анализатор достиг графического объекта, при первичном анализе обрабатываются функцией *generate_input*(), при вторичном анализе на базе ее результатов выполняется связывание фрагментов диаграммы, выполненной в терминах целевого языка, применением функции *stick_connection_points*().

На рис. 16 представлены все возможные варианты отображений, а в таблице 10 описаны соответствующие им типы выбора продолжателя и применения транслирующих функций. На отображениях (см. рис. 16) полыми точками показаны направляющие точки, заштрихованными – точки, через которые анализатор достигает данного графического объекта, а заполненными точками – все остальные входящие и исходящие точки.

В последующих таблицах принято кодирование транслирующих функций. Значению f_{gi} в ячейках таблиц соответствует вызов функции *generate_input*(), f_{go} – *generate_output*(), f_{so} – *select_output*(), f_{scp} – *stick_connection_points*().

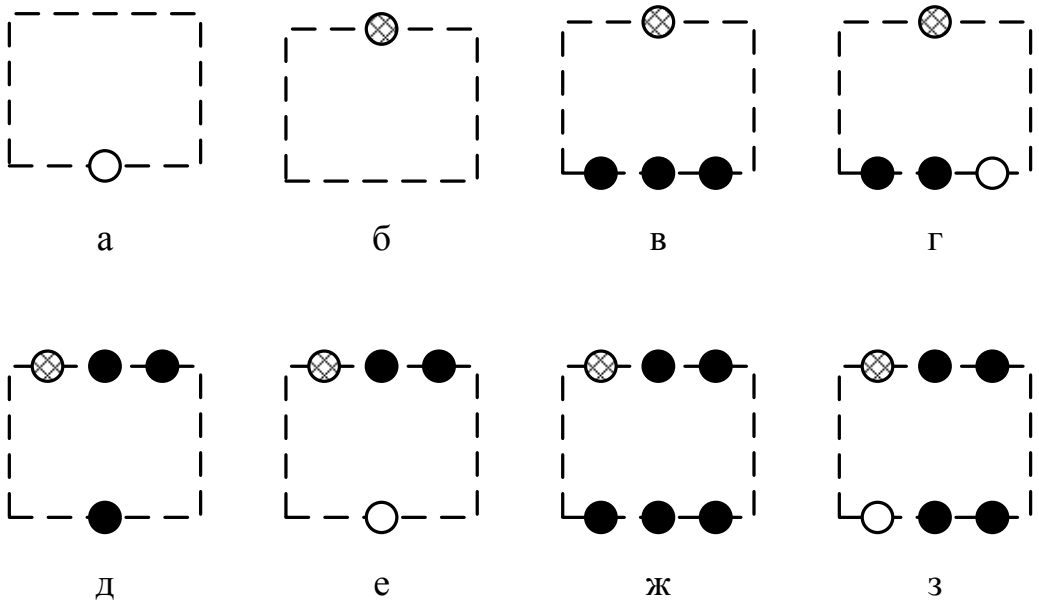


Рис. 16. Возможные типы отображений

Таблица 10. Описание возможных типов отображений

Описание отображения	**Транслирующие функции**	**Продолжатель**
с одним направленным выходом (рис. 16, а).	нет	направленный выход
с одним входом (рис. 16, б)	нет	нет, выбирается из числа связей-меток
с одним входом и несколькими ненаправленными выходами (рис. 16, в)	f_{go} – выбирает все выходы,	f_{so}, или выбирается из числа связей-меток
с одним входом и несколькими выходами, в том числе с направленным (рис. 16, г)	f_{go} – выбирает все выходы, кроме направленного	направленный выход
с несколькими входами и одним ненаправленным выходом (рис. 16, д)	f_{gi}, f_{go} – выбирает все выходы	нет, выбирается из числа связей-меток
с несколькими входами и одним направленным выходом (рис. 16, е)	f_{gi}	направленный выход
с несколькими входами и несколькими ненаправленными выходами (рис. 16, ж)	f_{gi}, f_{go} – выбирает все выходы	f_{so}, или выбирается из числа связей-меток
с несколькими входами и несколькими выходами, в том числе с направленным (рис. 16, з)	f_{gi}, f_{go} – выбирает все выходы, кроме направленного	направленный выход

Для построения табличной формы RVTg-грамматики для базового языка ПГС и целевого языка сетей Петри можно воспользоваться минимизированной табличной формой для RVTt-грамматики из таблицы 9.

В заимствованной таблице необходимо переработать операции над внутренней памятью целевого языка и назначить соответствующие отображения квазитермам. Транслирующие функции назначаются отображениям автоматически в соответствии с таблицей 10.

Внутренняя память целевого языка состоит из:

- четырех магазинов для хранения информации о переходах, входящих в состав отображений графических объектов (логического условия (11m), объединения взаимоисключающих ветвей (12m), распараллеливания (13m) и слияния (14m)), из которых будут следовать связи-метки;
- двух лент для хранения информации о переходах, входящих в состав отображений уже проанализированных графических объектов – объединениях взаимоисключающих ветвей (t(11)) и слияниях (t(12)).

Для пары языков ПГС и сетей Петри операции над внутренней памятью схожи. Элементы памяти целевого языка (сети Петри) хранят указатели на переходы, с которых начинается или завершается любое из отображений транслируемых графических объектов (см. рис. 17). Так, например, при первичном анализе логического условия необходимо запомнить переход, из которого исходит связь-метка (заполненная точка на рис. 17, в), для этого используется операция вида $W_1(t^{11m})$. При первичном анализе слияния (см. рис. 17, з) необходимо запомнить переход, из которого исходит связь-метка и не анализированные входящие переходы, в данной случае это все один переход, а операция над внутренней памятью выглядит так $W_1((k-1)^{t(12)}, t^{14m})$. Причем t обозначает отображение в общем, как будто это абстрактный объект с точками, но в память записывается ссылка на конкретный переход, $(k-1)$ обозначает количество не анализированных переходов.

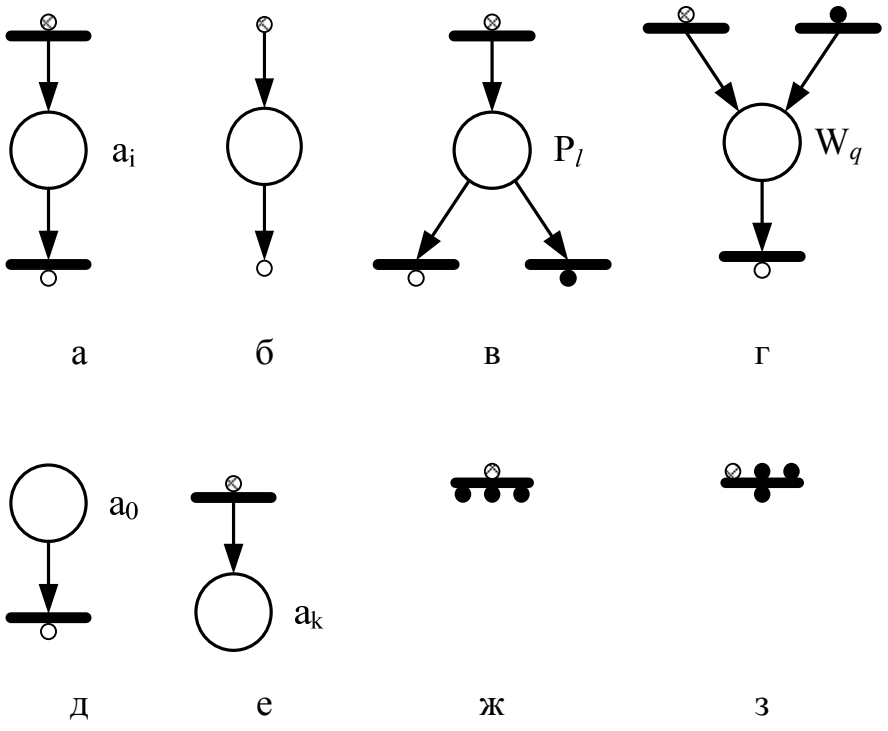


Рис. 17. Отображения в терминах сетей Петри для элементов языка ПГС: действие (а), связь (б), условие (в), объединение взаимоисключающих ветвей (г), начало (д), конец (е), распараллеливание (ж), слияние (з)

Результирующая табличная форма RVTg-грамматики для языков ПГС и сетей Петри представлена в таблице 11. Последняя колонка таблицы содержит сноски на отображения в терминах целевого языка, представленные на рис. 17.

Таблица 11. RVTg-грамматика для трансляции ПГС в сети Петри

№ п/п	**Комплекс**	**Квазитерм**	**Приемник**	**RVTg-отношения**		**Отображение**
				базовый язык	**целевой язык**	
1	r_0	A_0	r_1	$\oslash$		д
2	r_1	rel	r_3	$\oslash$		б
3	r_2	$label_P$	r_3	$W_2(b^{1m})$	$W_2(b^{11m})$	б
4		$label_W$	r_3	$W_2(b^{2m})$	$W_2(b^{12m})$	б
5		$label_R$	r_3	$W_2(b^{3m})$	$W_2(b^{13m})$	б

Окончание табл.11

№ п/п	Комп-лекс	Квазитерм	При-емник	RVTg-отношения		Отобра-жение
				базовый язык	целевой язык	
6		$label_L$	r_3	$W_2(b^{4m}) / W_3(m^{t(2)} == k^{t(3)})$	$W_2(b^{14m})$	б
7		no_label	r_k	*	* *	
8	r_3	A	r_1	∅		а
9		$P \multimap$	r_1	$W_1(t^{1m})$	$W_1(t^{11m})$	в
10		W	r_2	$W_1(1^{t(1)}, t^{2m}) / W_2(e^{t(1)})$	$W_1(1^{t(11)}, t^{12m})$	г
11		$\breve{W}$	r_2	$W_1(2^{t(1)}) / W_2(1^{t(1)})$	$W_1(0^{t(11)}) / W_2(1^{t(11}$	
12		R	r_1	$W_1(t^{3m^{(k-1)}}) / W_3(k > 1)$	$W_1(t^{13m^{(k-1)}})$	ж
13		L	r_2	$W_1(1^{t(2)}, k^{t(3)}, t^{4m}) / W_2(e^{t(2)})$	$W_1((k-1)^{t(12)}, t^{14m}$	з
14		$\breve{L}$	r_2	$W_1(inc(m^{t(2)})) /$ $W_3(m^{t(2)} < k^{t(3)})$	$W_1(dec(m^{t(12)}))$	
15		A_k	r_2	∅		е
16	r_k			∅		

По завершению анализа во внутренней памяти целевого языка магазины должны быть пусты, а все ячейки лент должны содержать символ «0». Проверка состояния памяти описывается операцией обозначенной символом $* *$.

Операция $*$:

$$* = W_2(e^{1m}) \,\&\&\, W_2(e^{2m}) \,\&\&\, W_2(e^{3m}) \,\&\&\, W_2(e^{4m}) \,\&\&\, W_2(2^{\alpha_i(1)}) \,\&\&$$
$$W_3(m^{\alpha_i(2)} > 2) \,\&\&\, W_3(m^{\alpha_i(2)} == k^{\alpha_i(3)})$$

$$* = W_2(e^{11m}) \,\&\&\, W_2(e^{12m}) \,\&\&\, W_2(e^{13m}) \,\&\&\, W_2(e^{14m})$$
$$W_3(m^{\alpha_i(11)} == 0) \,\&\&\, W_3(m^{\alpha_i(12)} == 0).$$

Для наглядности преобразования диаграмм базового языка в диаграммы целевого языка в таблице 12 представлена пошаговая трансляция ПГС в сети Петри. Каждая строка таблицы соответствует строке с тем же номером в таблице 11, показывает состояние внутренней памяти для целевого языка, используемые на данном шаге транслирующие функции (столбец «Транслирующая функция») и назначаемое отображение (столбец «Отображение»). Исходная диаграммы показана на рис. 17, а результат трансляции на рис. 18.

Таблица 12. Пошаговая трансляция ПГС в сеть Петри

№	Комплекс	Квазитерм	Состояние памяти						Транслир. функция	Отобра-жение
			M11	M12	M13	M14	L11	L12		
(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)
1	r_0	A_0								д
2	r_1	rel								б
3	r_3	R			t_2, t_2				f_{go}, f_{so}	ж
4	r_1	rel			t_2, t_2					б
5	r_3	W		t_5	t_2, t_2		1^{t_4}		f_{gi}, f_{go}	г
6	r_2	$label_W$			t_2, t_2		1^{t_4}			б
7	r_3	P	t_8		t_2, t_2		1^{t_4}		f_{go}	в
8	r_1	rel	t_8		t_2, t_2		1^{t_4}			б
9	r_3	L	t_8		t_2, t_2	t_9	1^{t_4}	2^{t_9}	f_{gi}, f_{go}	з
10	r_2	$label_P$			t_2, t_2	t_9	1^{t_4}	2^{t_9}		б
11	r_3	A			t_2, t_2	t_9	1^{t_4}	2^{t_9}		а
12	r_1	rel			t_2, t_2	t_9	1^{t_4}	2^{t_9}		б
13	r_3	$\breve{W}$			t_2, t_2	t_9	0^{t_4}	2^{t_9}	f_{scp}	
14	r_2	$label_R$			t_2	t_9	0^{t_4}	2^{t_9}		б
15	r_3	P	t_{14}		t_2	t_9	0^{t_4}	2^{t_9}	f_{go}	в
16	r_1	rel	t_{14}		t_2	t_9	0^{t_4}	2^{t_9}		б
17	r_3	A	t_{14}		t_2	t_9	0^{t_4}	2^{t_9}		а
18	r_1	rel	t_{14}		t_2	t_9	0^{t_4}			б
19	r_3	W	t_{14}	t_{19}	t_2	t_9	$0^{t_4}, 1^{t_{18}}$	2^{t_9}	f_{gi}, f_{go}	г
20	r_2	$label_P$		t_{19}	t_2	t_9	$0^{t_4}, 1^{t_{18}}$	2^{t_9}		б
21	r_3	P	t_{22}	t_{19}	t_2	t_9	$0^{t_4}, 1^{t_{18}}$	2^{t_9}	f_{go}	в

Окончание табл.12

№	Комплекс	Квазитерм	Состояние памяти						Транслир. функция	Отобра-жение
			M11	M12	M13	M14	L11	L12		
22	r_1	rel	t_{22}	t_{19}	t_2	t_9	$0^{t_4}, 1^{t_{18}}$	2^{t_9}		Б
23	r_3	$\breve{W}$	t_{22}	t_{19}	t_2	t_9	$0^{t_4}, 1^{t_{18}}$	2^{t_9}	f_{scp}	
24	r_2	$label_P$		t_{19}	t_2	t_9	$0^{t_4}, 1^{t_{18}}$	2^{t_9}		б
25	r_3	A		t_{19}	t_2	t_9	$0^{t_4}, 1^{t_{18}}$	2^{t_9}		а
26	r_1	rel		t_{19}	t_2	t_9	$0^{t_4}, 1^{t_{18}}$	2^{t_9}		б
27	r_3	W		t_{19}, t_{27}	t_2	t_9	$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	2^{t_9}	f_{gi}, f_{go}	г
28	r_2	$label_W$		t_{27}	t_2	t_9	$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	2^{t_9}		б
29	r_3	$\breve{W}$		t_{27}	t_2	t_9	$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	2^{t_9}	f_{scp}	
30	r_2	$label_W$			t_2	t_9	$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	2^{t_9}		б
31	r_3	$\breve{L}$			t_2	t_9	$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	1^{t_9}	f_{scp}	
32	r_2	$label_R$				t_9	$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	1^{t_9}		б
33	r_3	A				t_9	$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	1^{t_9}		а
34	r_1	rel				t_9	$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	1^{t_9}		б
35	r_3	$\breve{L}$				t_9	$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	0^{t_9}	f_{scp}	
36	r_2	$label_L$					$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	0^{t_9}		б
37	r_3	A_k					$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	0^{t_9}		е
38	r_2	no_label					$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	0^{t_9}		
39	r_k						$0^{t_4}, 0^{t_{18}}, 1^{t_{26}}$	0^{t_9}		

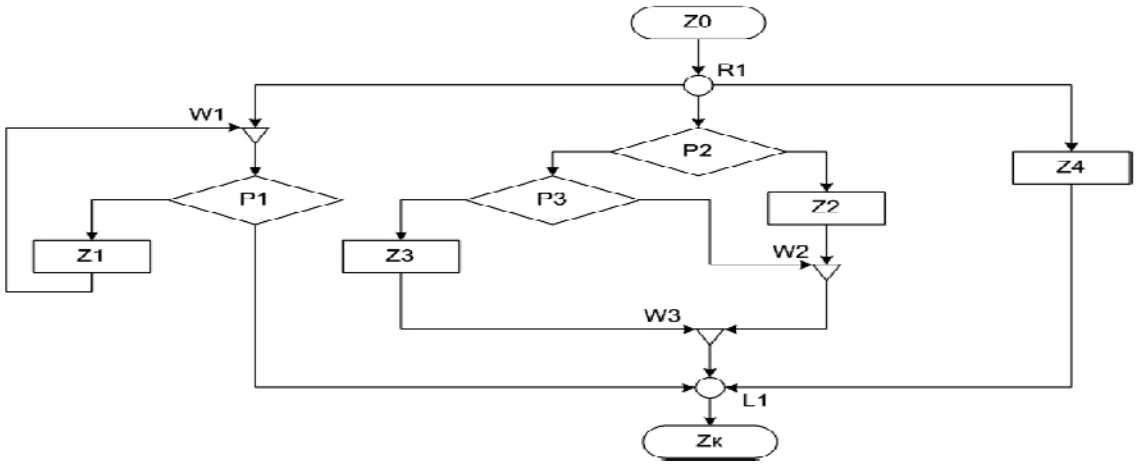


Рис. 17. Пример ПГС

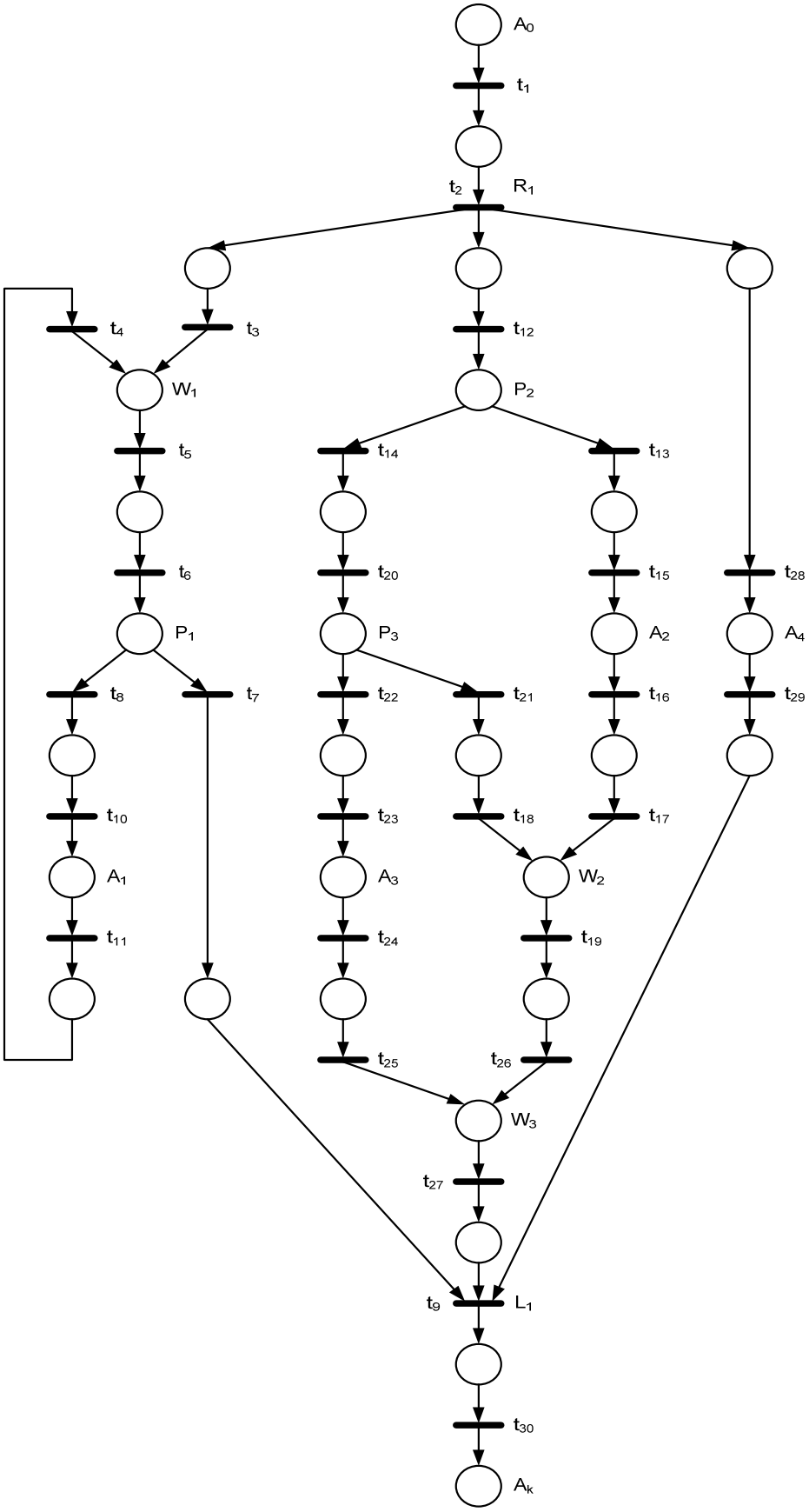


Рис. 18. Диаграмма к рис. 17 в терминах сетей Петри

4. Метатрансляция графических языков потоков проектных работ

Метакомпилятор используется на этапе создания схемы RV-грамматики. Основная цель его применения – упростить работу по составлению правил грамматики, сохраняя эффективность парсинга диаграммы. На входе метакомпилятор получает контекстно-свободное описание диаграммного языка, на выходе выдает описание автомата RV-грамматики в формате XML.

Метакомпилятор позволяет формализовать процесс построения RV-грамматики из контекстно-свободного описания и выявить условия, при которых построение проблематично, а также преобразовывать диаграммные языки, описанные другими грамматиками, в RV-грамматику и проводить сравнительные тесты анализаторов на одинаковых задачах.

Процесс метакомпиляции состоит из несколько этапов.

1. Лексический разбор описания диаграммного языка.
2. Синтаксический разбор и построение дерева разбора.
3. Анализ дерева разбора и построение промежуточной структуры грамматики в терминах языка спецификаций.
4. Трансляция – преобразование внутреннего представления в термины RV-грамматики.
5. Оптимизация и минимизация.
6. Сохранение таблиц RV-грамматики в формате XML.

В процессе анализа описания диаграммного языка ППР метакомпилятор формирует набор паросочетаний между терминальными символами языка, строит таблицу переходов автомата RV-грамматики и генерирует набор операций с памятью для контроля соответствия вложенности нетерминальных символов и блоков с количеством входов или выходов больше одного.

Отделение описания RV-грамматики от ее интерпретатора позволяет использовать полученный XML-файл на разных платформах вне

зависимости от того, с каким приложением интегрирован использующийся интерпретатор.

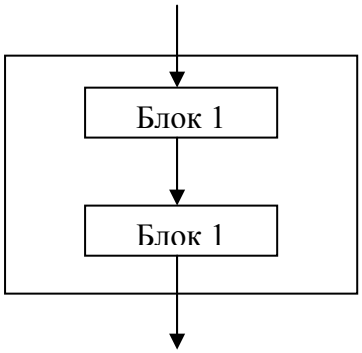
Рис. 19. Структура метакомпилятора диаграммных языков

Метакомпилятор представляет собой консольное приложение, написанное на C# с использованием GOLD Parser Builder для разбора языка метаописаний. Структура взаимодействия его компонент в составе проекта показана на рис. 19.

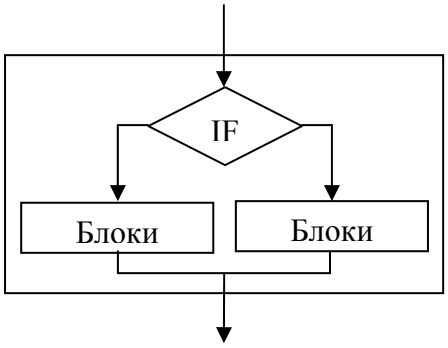
Описание спецификаций диаграммных языков состоит из правил, содержащих наименование нетерминального символа, следующего после зарезервированного слова Rule, перечисление набора терминальных и нетерминальных символов, входящих в правило, а также описание соотношений между терминальными и нетерминальными символами внутри правила и во внешнюю среду.

В качестве примера построим с помощью метакомпилятора RV-грамматику языка блок-схем ППР. Графические конструкции блок-схем и их описания показаны на рис. 20. Рамка, в которую заключены

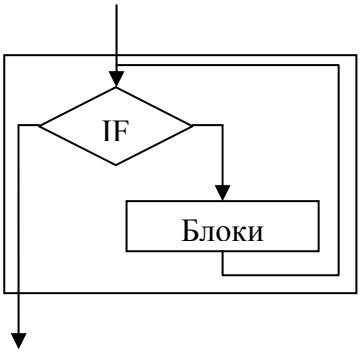
составляющие правил, обозначает границы нетерминального символа и иллюстрирует соответствие входов и выходов.



Rule Блоки
Consist of Блок Блок1, Блок Блок2
Internal Relationships: Блок1.Вход = Блок2.Выход
External Relationships: Блоки.Вход = Блок1.Вход, Блоки.Выход = Блок2.Выход



Rule Блоки
Consist of Блоки Блоки1, Блоки Блоки2, Условие
Internal Relationships: Условие.Выход1 = Блоки1.Вход, Условие.Выход2 = Блоки2.Вход
External Relationships: Блоки.Вход = Условие.Вход, Блоки.Выход = Блоки1.Выход, Блоки.Выход = Блоки2.Выход

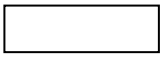
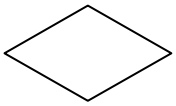
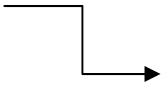


Rule Блоки
Consist of Блоки Блоки1, Условие
Internal Relationships: Условие.Выход2 = Блоки1.Вход, Условие.Вход = Блоки1.Выход
External Relationships: Блоки.Вход = Условие.Вход, Блоки.Выход = Условие.Выход1

Рис. 20. Конструкции блок-схем и их метаописания

Терминальные и квазитерминальные символы генерируются на основе приведенных правил. Для ветвления после анализа внешних выходов создаются дополнительные символы объединения взаимоисключающих ветвей W и W!. По правилу описания цикла создается квазитерминальный символ P! Ниже в таблице 13 приведено описание квазитерминальных символов.

Таблица 13. Терминальные и квазитерминальные символы блок-схем

Символ	**Квазитерминальный символ**	**Примечание**
	A	Блок
	P	Ветвление
	P!	Уже анализированное ветвление
	C	Объединение взаимоисключающих ветвей
	C!	Уже анализированное объединение ветвей
	R	Связь
	PR	Связь, исходящая из логического условия
	CR	Связь, исходящая из объединения ветвей
	0	Отсутствие связей меток

В результате получаем автоматную RV-грамматику (рис. 21). W^1 обозначает операцию записи в стек, W^2 – операцию чтения из стека.

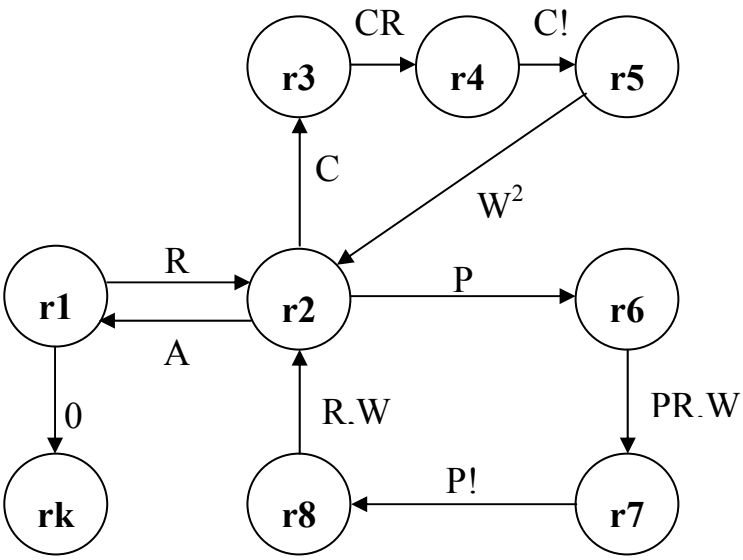


Рис. 21. Автоматная RV-грамматика блок-схем

На выходе метакопилятора получается RV-грамматика в формате XML, которая используется кодом RV-анализатора для разбора конкретных графических предложений.

<RVGrammar>
<Terms>
<Term name=”Блок” incount=”1” outcount=”1”><KvaziTerm>A</KvaziTerm></Term>
<Term name=”Условие” incount=”1” outcount=”2”>
<KvaziTerm name=”P” / ><KvaziTerm name=”P!” / ></Term>
<Term name=”Объединение” incount=”2” outcount=”1”>
<KvaziTerm name=”C” /><KvaziTerm name=”C!” / ></Term>
<Term name=”Связь” incount=”1” outcount=”1”>
<KvaziTerm name=”R” /><KvaziTerm name=”PR” /><KvaziTerm name=”WR” /></Term>
<GrammarTable>
<Source srcComplex=”r1”>
<Destination dstComplex=”r2” KvaziTerm=”R”></Destination>
<Destination dstComplex=”r0” KvaziTerm=”0”></Destination>
</Source>
<Source srcComplex=”r2”>
<Destination dstComplex=”r3” KvaziTerm=”C”></Destination>
<Destination dstComplex=”r1” KvaziTerm=”A”></Destination>
<Destination dstComplex=”r6” KvaziTerm=”P”> </Destination>

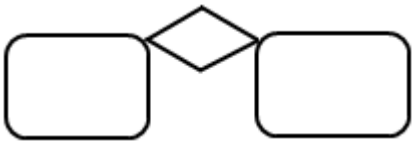
```
</Source>
<Source srcComplex=”r3”>
<Destination dstComplex=”r4” KvaziTerm=”CR”></Destination>
</Source>
<Source srcComplex=”r4”>
<Destination dstComplex=”r5” KvaziTerm=”C!”></Destination>
</Source>
<Source srcComplex=”r5”>
<Destination dstComplex=”r2” KvaziTerm=”R”><IF Type=”W2” value=”P” /></Destination>
</Source>
<Source srcComplex=”r6”>
<Destination dstComplex=”r7” KvaziTerm=”PR”><Do Type=”W1” value=”P” /></Destination>
</Source>
<Source srcComplex=”r7”>
<Destination dstComplex=”r8” KvaziTerm=”P!”></Destination>
</Source>
<Source srcComplex=”r8”>
<Destination dstComplex=”r2” KvaziTerm=”R” ><IF Type=”W2” value=”P” /></Destination>
</Source>
</GrammarTable>
</Terms>
</RVGrammar>
```

Таким образом, при создании собственной библиотеки графических примитивов (такая возможность предоставляется, например, в редакторах Visio, Dia) пользователь может описать графический язык ППР, задав метаописание графических конструкций.

5. Анализ ошибок в диаграммных моделях потоков проектных работ

Разработанные в пп. 1, 2 синтаксически-ориентированные методы контроля и анализа диаграммных языков ППР на основе нового класса RV-грамматик позволяют определять и нейтрализовать синтаксические и семантические ошибки диаграммных моделей ППР. Ниже, в табл. 14-22, приведены синтаксические и семантические ошибки диаграмматики UML (5 видов диаграмм, 17 типов синтаксических и 13 типов семантических ошибок) и IDEF0, IDEF3 (11 типов синтаксических и 18 типов семантических ошибок). Для контроля семантики декларативной составляющей ППР в виде текстовых атрибутов в продукции RV-грамматик вводится процедура онтологического анализа понятий предметной области. Для семантического анализа диаграммы классов UML используются операции со стековой и ленточной памятью, контролирующие подчиненное свойство наследования.

Таблица 14. Ошибки в диаграмме Activity

Синтаксические ошибки		
Отсутствие связи	• условное ветвление без связи с действием • условное ветвление без связи с распараллеливанием • распараллеливание без связи с действием • другие	
Ошибка передачи управления	• связь, входящая в другую связь	

Окончание табл. 14

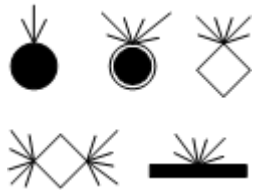
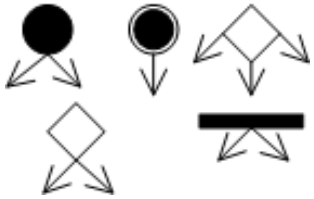
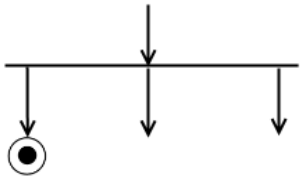
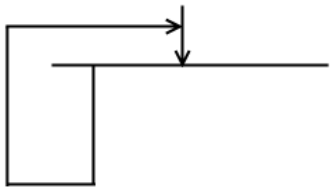
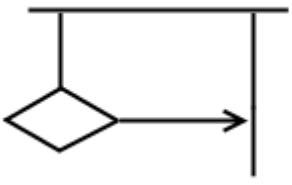
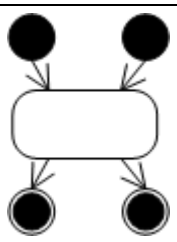
Ошибка кратности входов	• Узел начала-кратность – 0 • Узел конец – 1 • Условное ветвление – 1 • Слияние условных ветвей – 2 • Распараллеливание – 1 • Активность – 1	
Ошибка кратности выходов	• Узел начала – 1 • Узел окончания – 0 • Условное ветвление – 2 • Слияние условных ветвей – 1 • Слияние параллельных ветвей – 1 • Активность – 1	
Семантические ошибки		
Элемент конца в парал-лельной ветке	• не возможно завершить все параллельные ветви	
Бесконечный цикл	• возвращение параллельной ветви в распараллеливание	
Неверное переключение контекстов	• Условное ветвление должно заканчиваться в том же потоке, что и начиналось	
Диаграмма должна иметь конец и начало	• Начальный узел должен быть только один • Конечный узел должен быть только один	

Таблица 15. Ошибки в диаграмме классов

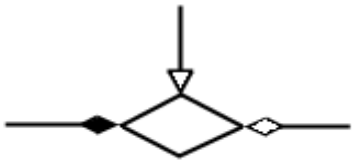
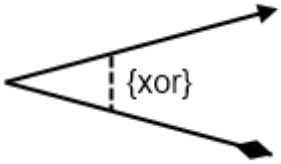
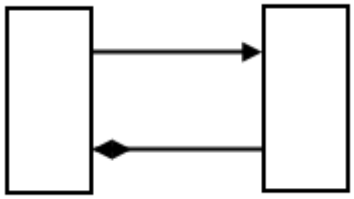
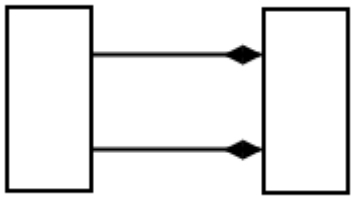
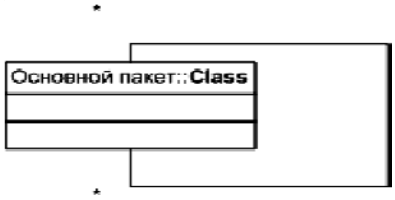
Синтаксические ошибки		
Отсутствие связи	• между классами нет связи	
Недопустимая связь	• в / из тернарной ассоциации входит / исходит не бинарная связь	
Ошибка связи	• вход связи любого типа в другую связь	
Исключающие связи неверного типа	• XOR объединение отношений не бинарного типа	
Семантические ошибки		
Кольцевые связи	• любой тип отношения между классом A и классом B, и одновременное какое-либо отношение между классами B и A	
Множественная однотипная связь	• включение класса A классом B, и одновременное расширение класса B классом A	
Ошибка ссылки класса на самого себя	• Класс не должен содержать себя как элемент прямо или косвенно	

Таблица 16. Ошибки в диаграмме последовательности

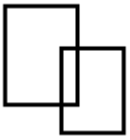
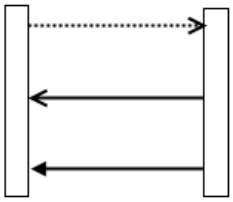
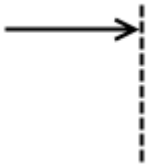
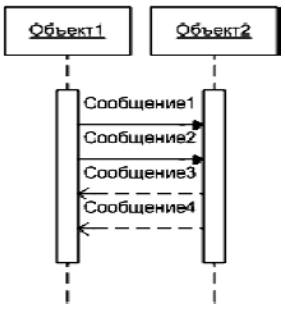
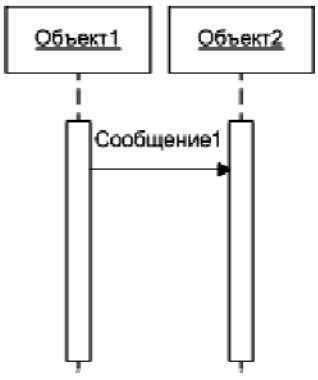
Синтаксические ошибки		
Отсутствие связи	• нерекурсивное пересечение фокусов управления	
Недопустимая связь	• связь вызова, направленная справа налево • связь возврата, направленная слева направо	
Вход связи в связь	• любой тип связи, входящий в другой тип связи	
Вызов, направленный в линию жизни	• вход линии вызова в линию жизни	
Семантические ошибки		
Синхронный вызов до получения ответа	• При синхронном вызове, нужно дождаться ответа перед следующим вызовом	
Стартовые и финальные события должны быть на одной Lifeline	• События должны начинаться и заканчиваться на одном объекте	

Таблица 17. Ошибки в диаграмме компонентов

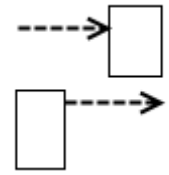
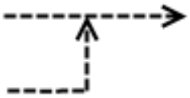
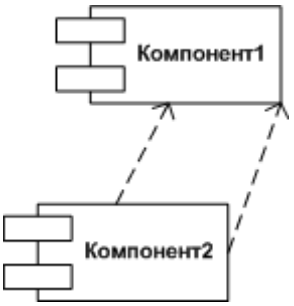
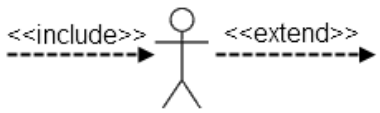
Синтаксические ошибки		
Не связанная связь	• Связь, исходящая из элемента, но не входящая никуда • Связь, входящая в элемент, но не исходящая из другого элемента	
Вход связи в связь	• связь, входящая в другую связь	
Семантические ошибки		
Интерфейс без реализации	• объявленный интерфейс без реализации	
Нарушение кратности зависимостей	• Зависимость между компонентами должна быть только одна	

Таблица 18. Ошибки в диаграмме Use Case

Синтаксические ошибки		
Отсутствие связи	• связь actor-case • связь actor-actor • связь case-case	
Недопустимая связь	• связь включения/расширения для actor • отношение ассоциации между классами	

Окончание табл. 18

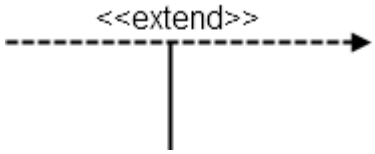
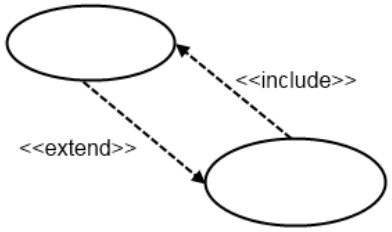
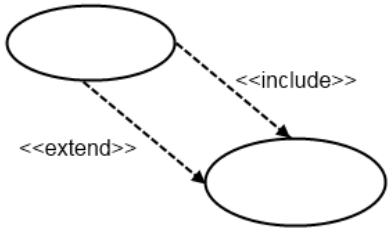
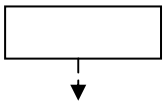
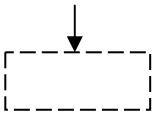
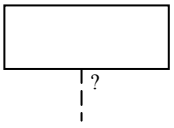
Синтаксические ошибки		
Вход связи в связь	• любой тип связи, входящий в другой тип связи	
Семантические ошибки		
Кольцевые связи	• включение класса А классом В и одновременное расширение класса В классом А	
Взаимоисключающие связи	• одновременное расширение и включение класса В классом А	

Таблица 19. Общие синтаксические ошибки всех графических нотаций IDEF

Ошибка	**Описание**
	Ожидалась связь
	Ожидался графический объект
	Некорректный тип связи

Окончание табл. 19

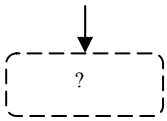
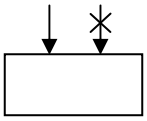
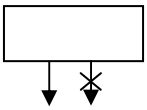
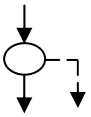
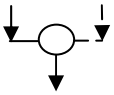
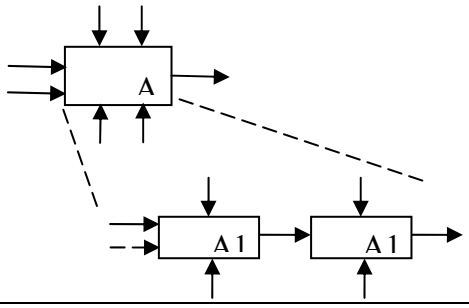
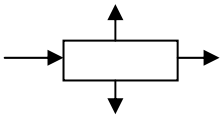
Ошибка	**Описание**
	Некорректный тип графического объекта
	Лишняя входящая связь в графический объект
	Лишняя исходящая связь из графического объекта
	Недостаточное количество исходящих связей из элемента с фиксированным минимальным количеством выходов
	Недостаточное количество входящих связей в объект с фиксированным минимальным количеством выходов
	Несоответствие граничных связей между родительской и дочерней диаграммой (показан пример для IDEF0)
	Геометрически неверное направление одной из связей (показан пример для IDEF0)
	Некорректное слияние стрелок (показан пример для IDEF0)

Таблица 20. Семантические ошибки IDEF0

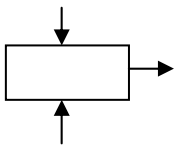
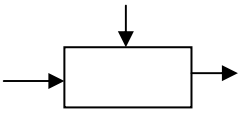
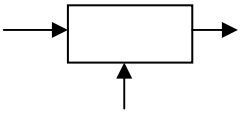
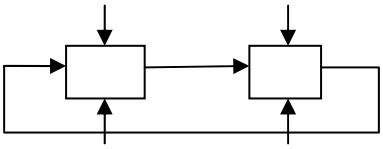
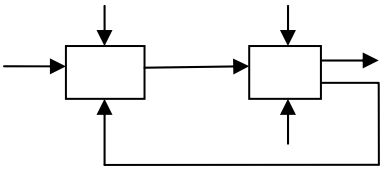
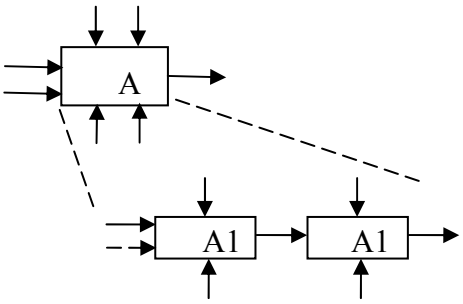
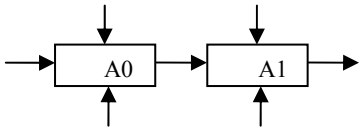
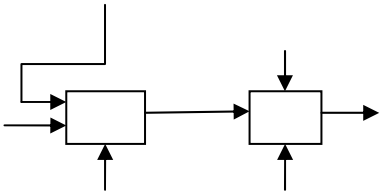
Графическое представление ошибки	**Комментарий**
	Отсутствие входных данных процесса
	Отсутствие механизма, действующего на процесс
	Отсутствие управления процессом
	Отсутствие входных/выходных данных диаграммы
	Блокировка процесса
	Несоответствие граничных связей между родительской и дочерней диаграммой
	Некорректная A0-диаграмма (диаграмма верхнего уровня)
	Нарушение семантики граничной связи

Таблица 21. Семантические ошибки IDEF1X

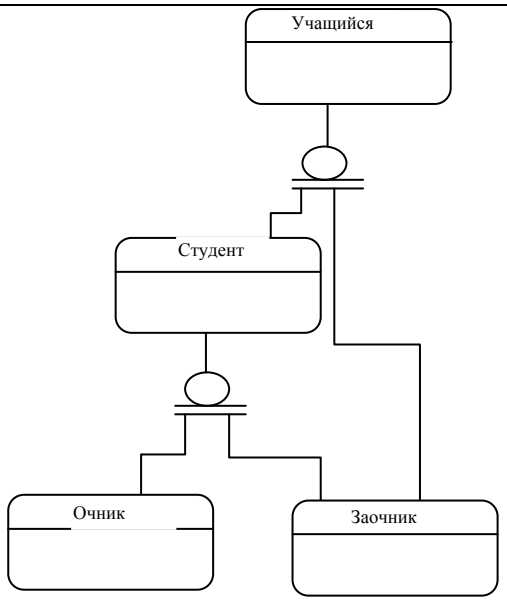
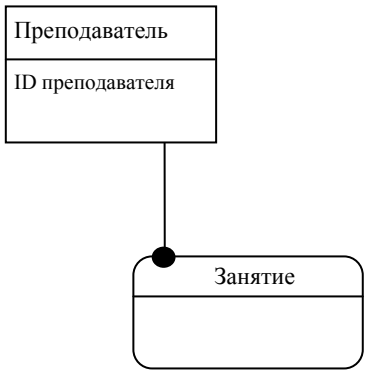
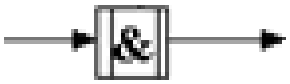
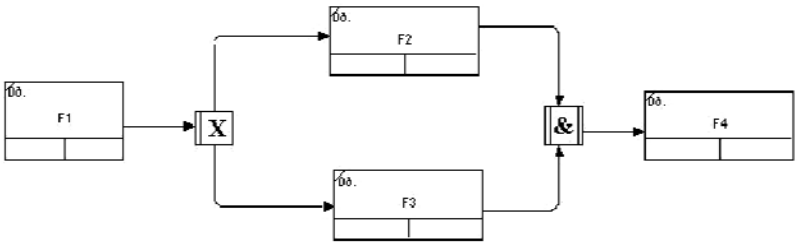
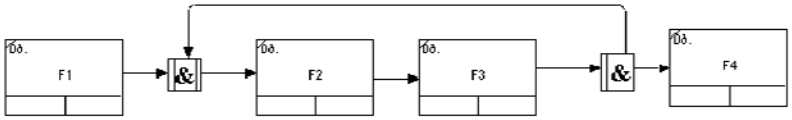
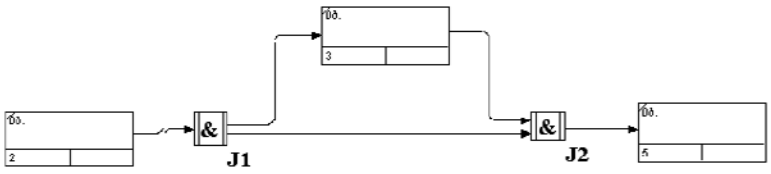
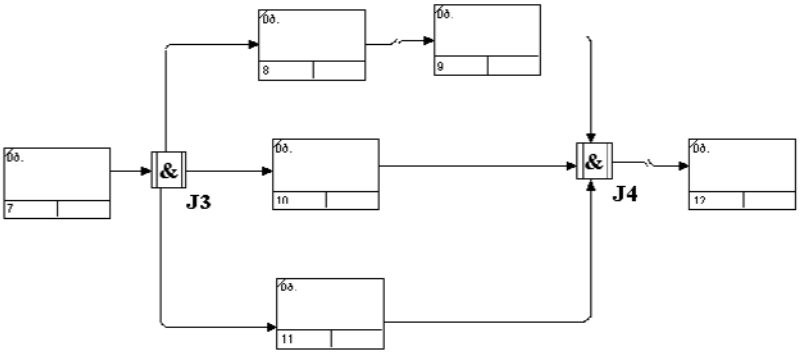
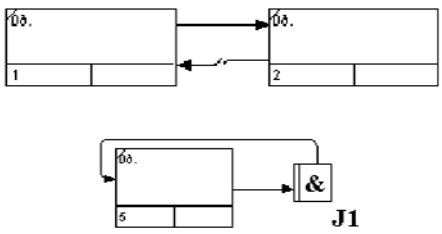
Графическое представление ошибки	**Комментарий**
	Ошибка множественного наследования
	Некорректный тип сущности (сущность «Занятие» – зависимая)
	Отсутствие внешнего ключа у зависимой сущности

Таблица 22. Семантические ошибки IDEF3

Графическое представление ошибки	Комментарий
	Нарушение семантики коннектора (коннектор должен содержать либо несколько выходов, либо несколько входов)
	Зависание (F4 никогда не выполнится)
	Дедлок (F4 никогда не выполнится)
	Рассинхронизация процессов (время выполнения одной ветви не равно времени выполнения другой)
	
	Зацикливание

Для анализа эффективности работы RV-анализаторов в группе из десяти магистрантов направления «Информатика и ВТ» был проведен учебный эксперимент, заключающийся в разработке UML и IDEF-диаграмм для 15 задач. Разработанные диаграммы содержали не более 20 вершин. RV-анализаторы показали 100% обнаружение ошибок. Анализ полученных с помощью генератора диаграмм с ошибками также показал высокий результат контроля (99,6%; 0,4% относятся к семантическим ошибкам, связанным с удаленным нарушением контекста взаимодействия ППР).

6. Темпоральная RVTI-грамматика, онтология и пример применения

Темпоральной автоматной RVTI-грамматикой языка L (G) называется упорядоченная восьмерка непустых множеств $G = \left(V, \Sigma, \tilde{\Sigma}, C, E, R, \tau, r_0\right)$, где $V = \left\{v_e,\ e = \overline{1.L}\right\}$ – вспомогательный алфавит (алфавит операций над внутренней памятью, представленный магазином или эластичной лентой); $\Sigma = \left\{a_n,\ n = \overline{1.T}\right\}$ – алфавит графических символов (объектов); $\tilde{\Sigma} = \left\{\widetilde{a_n},\ n = \overline{1.\tilde{T}}\right\}$ – квазитерминальный алфавит, являющийся расширением терминального алфавита $\sum$; $C = \left\{c, c = c + t_l \mid \exists t_0 = 0 \to c = 0\right\}$ – идентификатор часов (счетчик); $\tau = \left\{t_l \in [0; +\infty], l = \overline{1.K}\right\}$ – множество временных меток, причем $c \in [t_l; t_{l+1}]$; E – темпоральное отношение вида $\{c \sim t_l\}$, где переменная c (идентификатор часов), отношение $\sim \in \{=, \langle, \leq, \rangle, \geq\}$), описывающих условие наступление события t_l; $R = \left\{r_i,\ i = \overline{0,I}\right\}$ – схема грамматики G (множество имен комплексов продукций, причем каждый комплекс r_i состоит из подмножества P_{ij} продукций $r_i = \left\{P_{ij},\ j = \overline{1.J}\right\}$); $r_0 \in R$ – аксиома RVTI-грамматики (имя начального комплекса продукций), $r_k \in R$ – заключительный комплекс продукций. Продукция $P_{ij} \in r_i$ имеет вид $\tilde{a}_l \overset{\{W_\gamma(v_1, \ldots, v_n) \mid E\}}{\to} r_m$, где $W_\gamma\left(v_1, \ldots, v_n\right)$ – n-арное отношение, определяющее вид операции над внутренней памятью в зависимости от $\gamma = \{1, 2, 3\}$ (1 – запись, 2 – чтение, 3 – сравнение), $E = \varnothing$ при $c \neg \sim t_l$; $(\tilde{a}_l, t_l)$ – слова в виде пары квази-символа и временной метки; $r_m \in R$ – имя комплекса продукции-преемника. Язык данной грамматики

содержит слова вида $(\tilde{a}_l, t_l)$ при $E \neq \varnothing$ и $\tilde{a}_l$ при $E = \varnothing$, представляет трассу $\sigma = \left\{\tilde{a}_0, 0\right\} \to \left\{\tilde{a}_l, t_l\right\} \vee \tilde{a}_l \to \left\{\tilde{a}_k, t_K\right\}$.

Поскольку онтология является совокупностью схем описания предметной области и правил отнесения данных к этой предметной области, а выделение онтологии является описанием схемы предметной области, характеризующейся определенной логической структурой, то с использованием семантического анализа диаграмматических моделей и набора написанных правил осуществляется наполнение выделенной онтологии. Логическая форма представления онтологии структурирована, поэтому к наполняющим ее данным применима реляционная алгебра. Онтология представлена следующим образом:

$$O = (Class,\ Property,\ Relation,\ Axiom),$$

где *Class* – множество понятий (классов), определенных для конкретной предметной области; *Property* – множество свойств понятия; *Relation* – множество базовых отношений и семантических связей, определенных между понятиями в *Class*. Множеством базовых отношений являются: синонимия, разновидность чего-либо, часть чего-либо (*f*), экземпляр чего-либо, свойство чего-либо (*property_of*); *Axiom* – множество аксиом. Аксиома – это реальный факт или правило, определяющее причинно-следственную связь.

Для построения модели конструкторско-технологических потоков работ выделим одну из задач конструкторской подготовки производства: рассмотрим типовую процессную модель согласования конструкторской документации детально. Для построения модели спроектирован бизнес-процесс разработки и согласования КД, определены правила формирования комплекта КД, выделены задания и определены их исполнители. В качестве инструмента для проектирования модели потока

работ использовано специализированное программное обеспечение Workflow Designer Системы управления проектами работ (СУПР), разработанное РЦ Аскон-Волга. Построена последовательность прохождения заданий, разработаны скрипты для изменения состояний документов в процессе согласования и заполнения атрибутов согласования в Лоцман: PLM. На рис. 22 представлена модель потока работ разработки и согласования конструкторской документации (КД) на специализированном языке СУПР.

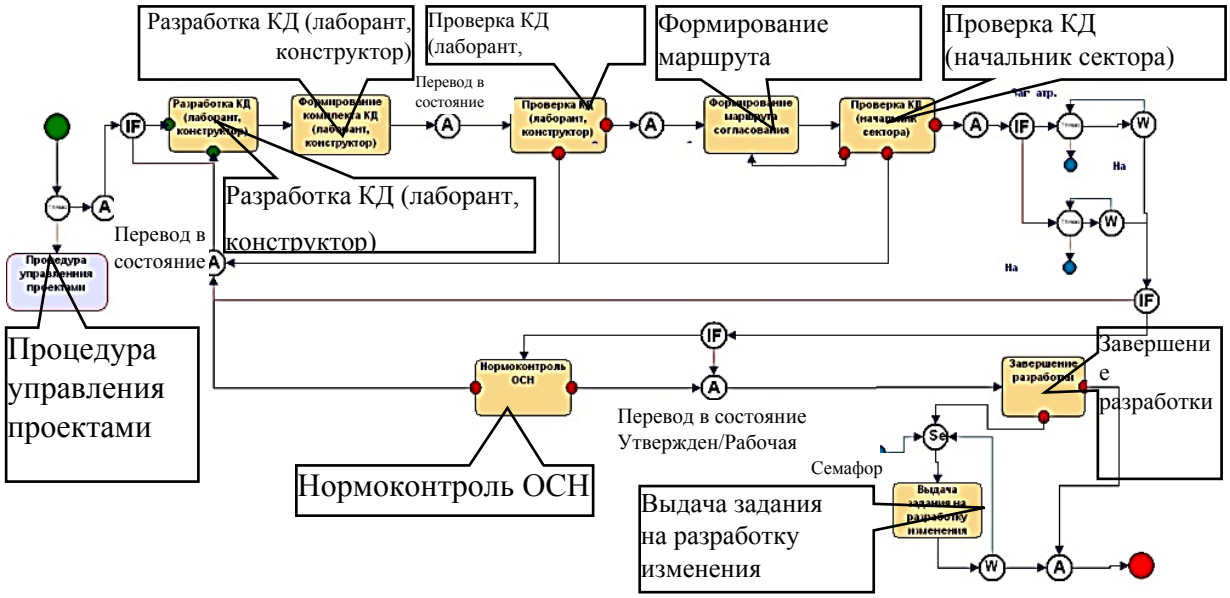


Рис. 22. Модель конструкторского согласования КД на специализированном языке компании АСКОН-Волга

Для визуализации потоков работ в виде диаграмм программных продуктов фирмой АСКОН разработан специализированный визуальный язык, позволяющий организовать вложенность процессов за счет нескольких элементов, представляющих собой свернутый подпроцесс. В языке поддерживается создание параллельных потоков. Контроль таких потоков возможен с помощью двух элементов «Событие» и «Семафор», которые работают в паре с элементом «Ожидание» и дополнительным типом связи «Синхродействие». Используется элемент «Фантом», который

позволяет соединять части диаграммы на разных уровнях вложенности. В таблице 23 показан ряд графических элементов языка, в таблице 24 – соответствие графических элементов нотации с квазитермами.

В таблице 25 приведена разработанная автоматная темпоральная RVTI-грамматика языка, позволяющая провести анализ структуры диаграмм указанного языка и выявить 1-19 классы ошибки, остальные 4 класса ошибки определяются с помощью онтологии.

Таблица 23. Элементы специализированного языка АСКОН

Название графического элемента	**Графический элемент нотации**	**Описание, особенности**
Процедура		Свернутый подпроцесс, который возможно вызывать многократно. Имеет только одну входящую связь типа «Перейти в процедуру»
Задание		Свернутый подпроцесс, действие в котором обязательно к выполнению пользователем
Итерация		Свернутый подпроцесс, выполнение которого требуется многократно
Вызов процедуры		Используется совместно с «Перейти в процедуру» и блоком «Процедура»
Создание потока		Используется совместно с «перейти в процедуру» и блоком «процедура»
Нетранзит		Операция, выполняемая пользователем
Скрипт (Автооперация)		
Ветвление		Имеет только две исходящие связи. True и false соответственно

Окончание табл. 23

Фантом		Позволяет соединять части диаграммы на разных уровнях вложенности. По сути является связью
Событие		Используется совместно с «Ожидание». Три дуги входящих, одна из них «Синхродействие», оповещающая о выполнении события.
Семафор		Используется совместно с «Ожидание». Три входящих, одна из них «Синхродействие», оповещающая о начале и завершении выполнения событий.
Активировать		Имеет две исходящих ветви, одна из них – «Синхродействие», оповещающая «Событие» об успешном завершении события
Ожидание		Имеет две исходящих ветви, одна из них «Синхродействие», следящая за статусом выполнения события. Пропускает поток только в случае успешного завершения события
Инкремент		Имеет две исходящих ветви, одна из них «Синхродействие», оповещающая «Семафор» о начале исполнения события
Декремент		Имеет две исходящих ветви, одна из них «Синхродействие», оповещающая «Семафор» о завершении исполнения события

Таблица 24. Соответствия графических элементов нотации с квазитермами

Название графического элемента нотации	**Квазитерм**	**Описание, особенности**
Процедура	vPR	Подграмматика для процедуры
Задание	vA	Подграмматика для задания
Итерация	vIT	Подграмматика для итерации
Точка входа	A0	
Точка выхода	Ak	Одна входящая связь
	Akm	Много входящих связей
	_Akm	Проанализированы не все входящие связи
	Akme	Проанализированы все входящие связи
Вызов процедуры	CL	
Создание потока	TH	
	THa	После входящей «Синхродействие»
Скрипт (автооперация)	SC	Одна входящая связь
	SCm	Много входящих связей
	_SCm	Проанализированы не все входящие связи
	SCme	Проанализированы все входящие связи
Ветвление	C	
	labelC	
Фантом	PHsp	Фантом с исходящей связью «Перейти в процедуру»
	PHep	Фантом с входящей связью «Перейти в процедуру»
	PHsa	Фантом с исходящей связью «Синхродействие»
	PHea	Фантом с входящей связью «Синхродействие»
	Phsai	Фантом с исходящей связью «Синхродействие», идущей от элемента «инкремент» или «активировать»

Окончание табл. 24

Название графического элемента нотации	**Квазитерм**	**Описание, особенности**
	Pheai	Фантом с входящей связью «Синхродействие», идущей от элемента «инкремент» или «активировать»
	Phsad	Фантом с исходящей связью «Синхродействие», идущей от элемента «декремент»
	Phead	Фантом с входящей связью «Синхродействие», идущей от элемента «декремент»
Событие	EV	Используется совместно с «ожидание». Три входящих, две из них «Синхродействие»
	EVa	Переход к элементу по входящей «Синхродействие»
Семафор	S	Используется совместно с «Ожидание». Три входящих, две из них «Синхродействие»
	Sa	Переход к элементу по входящей «Синхродействие»
Активировать	F	Две исходящих, одна из них «Синхродействие»
Ожидание	W	Две исходящих, одна из них «Синхродействие»
Инкремент	IN	Две исходящих, одна из них «Синхродействие»
Декремент	D	Две исходящих, одна из них «Синхродействие»
Перейти	rel	
Перейти, с условием «нет»	nrel	
Перейти в процедуру	prel	
Синхродействие	arel	
	airel	После элемента «Инкремент»
	adrel	После элемента «Декремент»
Возврат из подпроцесса	return	
	no_label	

Таблица 25. Темпоральная RVTI-грамматика языка АСКОН

Комплекс-источник	**Квазитерм**	**Комплекс-приемник**	**Операция с памятью**
r_0,t_0	A_0	r_3,t3	∅/E
r_1,t_1	return	r_2,t_4	$w_2(b^{4m})$
r_2,t_2	vA	r_1,t_1	$w_1(s^{1m},t^{4m})$, CALL vA/E
	vIT	r_1,t_1	$w_1(s^{1m},t^{4m})$, CALL vIT/E
	Ak	r_4,t_4	∅
	Akm	r_5,t_5	$w_1(1^{t(1)},i^{t(2)})/w_2(e^{t(1)})$/E
	_Akm	r_5,t_5	$w_1(inc(m^{t(1)}))/w_3(m^{t(1)}<k^{t(2)}-1)$,E
	Akme	r_4,t_4	$w_1(inc(m^{t(1)}))/w_3(m^{t(1)}=k^{t(2)}-1)$,E
	CL	r_6,t_6	$w_1(t^{4m})$
	TH	r_6,t_6	$w_1(1^{t(7)}, i^{t(8)}, t^{4m})$
	SC	r_3,t_3	∅
	SCm	r_5,t_5	$w_1(1^{t(3)},i^{t(4)})/w_2(e^{t(3)})$,E
	_SCm	r_5,t_5	$w_1(inc(m^{t(3)}))/w_3(m^{t(3)}<k^{t(4)}-1)$,E
	SCme	r_3,t_3	$w_1(inc(m^{t(3)}))/w_3(m^{t(3)}=k^{t(4)}-1)$,E
	C	r_7,t_7	$w_1(t^{2m})$
	EV	r_3,t_3	$w_1(0^{t(5)}, 0^{t(9)},0^{t(11)})/w_2(e^{t(5)})$,E
	S	r_3,t_3	$w_1(0^{t(6)}, 0^{t(10)},0^{t(12)})/w_2(e^{t(6)})$,E
	F	r_{11},t_{11}	$w_1(t^{3m})$
	W	r_9,t_9	$w_1(t^{3m})$
	IN	r_{11},t_{11}	$w_1(t^{3m})$
	D	r_{12},t_{12}	$w_1(t^{3m})$
r3,t3	rel	r_2,t_2	∅
r4,t4	no_label	r_{17},t_{17}	*
r_5,t_5	labelC	r_2,t_2	$w_2(b^{2m})$
r_6,t_6	prel	r_{13},t_{13}	∅
r_7,t_7	nrel	r_2,t_2	∅
r_8,t_8	PHsp	r_6,t_6	∅
r_9,t_9	arel	r_{14},t_{14}	∅

Окончание табл. 25

Комплекс-источник	**Квазитерм**	**Комплекс-приемник**	**Операция с памятью**
r_{10},t_{10}	PHsa	r_9,t_9	∅
r_{11},t_{11}	airel	r_{15},t_{15}	∅
r_{12},t_{12}	adrel	r_{16},t_{16}	∅
r_{13},t_{13}	vPR	r_1,t_1	$w_1(s^{1m})$, CALL(vPR)/E
	PHep	r8,t8	∅
r_{14},t_{14}	THa	r_2,t_2	$w_1(inc(m^{t(7)}))/w_3(m^{t(7)}<k^{t(8)})$,E
	PHea	r_{10},t_{10}	∅
	EVa	r_2,t_2	$w_1(1^{t(9)})$, $w_2(b^{3m})$
	Sa	r_2,t_2	$w_1(1^{t(10)})$, $w_2(b^{3m})$
r_{15},t_{15}	EVa	r_2,t_2	$w_1(inc(m^{t(5)}), 1^{t(11)})$, $w_2(b^{3m})$
	Sa	r_2,t_2	$w_1(inc(m^{t(6)}), 1^{t(12)})$, $w_2(b^{3m})$

По экземпляру диаграмматической модели (рис. 22) можно построить онтологию, классы которой являются словами (концептами) и имеют следующий вид: $Class_0 = \left(\tilde{a}_k, t_0\right), \ldots, Class_k = \left(\tilde{a}_k, t_k\right)$, где пара $\left(\tilde{a}_i, t_i\right)$ является темпоральным словом. Классы имеет свойства, которые представлены следующим образом: $Property = \{Name, OrientationTime, ProcessingTime\}$, где $Name$ – имя поля (наследуется от имени нотации диаграмматической модели); $OrientationTime$ – время начала потока; $ProcessingTime$ – длительность потока. Экземпляры записи $Property$ для классов, извлеченные из темпоральных слов с квазитермами vPR и vA, имеют следующий вид:

$$\text{Property}_{vPR} = \{\textit{Процедура управления проектами}, 0, 1\},$$

$$\text{Property}_{vA1} = \{\textit{Разработка КД}\,(\textit{лаборант}, \textit{конструктор}), 1, 1\},$$

$$\text{Property}_{vA2} = \{\textit{Формирование комплекта КД}\,(\textit{лаборант}, \textit{конструктор}), 2, 1\},$$

$$\text{Property}_{vA3} = \{\textit{Проверка КД}\,(\textit{лаборант}, \textit{конструктор}), 3, 1\},$$

$$\text{Property}_{vA4} = \{\textit{Формирование маршрута согласования}, 4, 1\},$$

$$\text{Property}_{vA5} = \{\textit{Проверка КД}\,(\textit{начальник сектора}), 5, 1\},$$

$$\text{Property}_{vA6} = \{\textit{Нормоконтроль ОСН}, 6, 1\},$$

$$\text{Property}_{vA7} = \{\textit{Завершение разработки}, 7, 1\},$$

$$\text{Property}_{vA8} = \{\textit{Выдача задания на разработку изменения}, 8, 1\}.$$

Разработанная авторская компьютерная программа [88] выполнила контроль и анализ диаграмматической модели (рис. 22) на наличие ошибок с 1 по 23 класс и вывела результат, что данная диаграмматическая модель не имеет ошибок из указанного перечня.

7. Денотативная и сигнификативная семантика диаграмматических моделей визуальных языков РЦ АСКОН-Волга, BPMN и eEPC

Денотативная семантика любого визуального языка представлена денотатами в виде графических объектов (слов) [89-91]. Под денотатом слова в теории визуальных языков понимается экземпляр класса с конкретными значениями свойств, характеризующих принадлежность слова к предмету. Общая структура экземпляра класса графического слова представлена в листинге 1.

```
Класс имя класса=start
начало
   свойство 1=значение 1
   свойство 2=значение 2
   свойство 3=значение 3
   свойство n=значение n
конец
```

Листинг 1. Обобщенная структура денотата графического слова

Структурные единицы параметров (свойства) графического слова представляют сигнификат этого слова, а сами свойства объединены в класс (Листинг 2).

```
Класс имя класса: текст
начало
   свойство 1: тип 1
   свойство 2: тип 2
   свойство 3: тип 3
   свойство n: тип 4
конец
```

Листинг 2. Обобщенная структура сигнификата графического слова

Описание денотатов и сигнификатов на теоретико-множественном языке имеет следующий вид:

NameOfDenotate=(ValueOfProperty_1,ValueOfProperty_2, ValueOfProperty_3, …, ValueOfProperty_N),

где NameOfDenotate – наименование денотата (экземпляра класса) графического слова;

ValueOfProperty_1 – значение первого свойства графического слова;

ValueOfProperty_2 – значение второго свойства графического слова;

ValueOfProperty_3 – значение третьего свойства графического слова;

ValueOfProperty_N – значение N-го свойства графического слова.

Significatum=(Property1, Property2, Property3, …, PropertyN),

где

Significatum – наименование сигнификата (класса) графического слова;

Property1 – структура первого свойства графического слова;

Property2 – структура второго свойства графического слова;

Property3 – структура третьего свойства графического слова;

PropertyN – структура N-го свойства графического слова.

Отличием сигнификата от денотата является то, что денотат есть экземпляр сигнификата, т. е. свойства (структура) денотата имеют конкретные значения. Далее в таблицах 26, 27, 28 представлены денотативная и сигнификативная семантики диаграмматических моделей гибридных динамических потоков проектных работ визуальных языков РЦ АСКОН Волга, BPMN, eEPC [41, 43, 64, 98, 99].

Таблица 26. Денотативная и сигнификативная семантика визуального языка РЦ АСКОН-Волга

Графическое слово	Понятие	Класс	Свойства класса	Значения свойства класса
	Свернутый подпроцесс, который возможно вызывать многократно. Имеет только одну входящую связь типа «Перейти в процедуру»	Процедура	Вход Алгоритм Выход	Вход=поток_работ Алгоритм=исходный_код Выход=поток_работ
	Свернутый подпроцесс, действие обязательно к выполнению пользователем	Задание	Вход Текст_задания Выход	Вход=поток_работ Текст_задания= скрипт_задания Выход=поток_работ
	Свернутый подпроцесс, выполнение которого требуется многократно	Итерация	Алгоритм	Алгоритм=исходный_код
	Используется совместно с «перейти в процедуру» и блоком «процедура»	Вызов процедуры	Адрес_ вызова _процедуры	Адрес_ вызова _процедуры=число
	Используется совместно с «перейти в процедуру» и блоком «процедура»	Создание потока	Имя Адрес_процедуры	Имя=название_потока Адрес_процедуры=число
	Операция, выполняемая пользователем	Нетранзит	код_операции адрес_операдна1 адрес_операнда2	код_операции=число адрес_операнда1=число адрес_операнда2=число
	Автоматизированное (автоматическое) выполнения операций	Скрипт (Авто-операция)	Алгоримт	Алгоритм=исходный_код
	Имеет только две исходящие связи. True и False соответственно	Ветвление	Условие	Условие=исходный_код
	Позволяет соединять части диаграммы на разных уровнях вложенности. По сути является связью	Фантом	Из_уровня К_уровню	Из_уровня=номер_уровня К_уровню=номер_уровня

Окончание табл. 26

Графическое слово	Понятие	Класс	Свойства класса	Значения свойства класса
	Используется совместно с «Ожидание». Три входящих, одна из них «Синхродействие», оповещающая о выполнении события.	Событие	Время	Время=число
	Используется совместно с «Ожидание». Три входящих, одна из них «Синхродействие», оповещающая о начале и завершении выполнения событий.	Семафор	Начало Завершение	Начало=число Завершение=число
	Имеет две исходящих ветви, одна из них «Синхродействие», оповещающая «Событие» об успешном завершении события	Активи-ровать	Выход	Выход=логическое_зна-чение
	Имеет две исходящих ветви, одна из них «Синхродействие», следящая за статусом выполнения события. Пропускает поток только в случае успешного завершения события	Ожидание	Длительность	Длительносьт=число
	Имеет две исходящих ветви, одна из них «Синхродействие», оповещающая «Семафор» о начале исполнения события	Инкремент	С По Шаг	С=число По=число Шаг=число
	Имеет две исходящих ветви, одна из них «Синхродействие», оповещающая «Семафор» о завершении исполнения события	Декремент	От До Шаг	От=число До=число Шаг=число

Таблица 27. Денотативная и сигнификативная семантика визуального языка BPMN

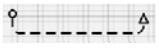
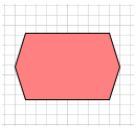
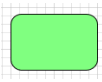
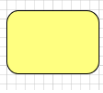
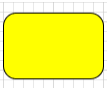
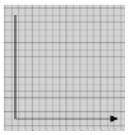
Графическое слово	Понятие	Класс	Свойства класса	Значения свойства класса
	Не имеет входящих потоков	Стартовое событие	Выход	Выход=поток_работ
	Лишь один входящий и исходящий поток	Промежуточное событие	Вход Операция Выход	Вход=поток_работ Операция=алгоритм Выход=поток_работ
	Может быть триггерным	Промежуточное событие «Сообщение»	Вход Событие Выход	Вход=поток_работ Событие=алгоритм Выход=поток_работ
	Не имеет исходящих потоков	Конечное событие	Вход	Вход=поток_работ
	Входящие потоки	Действие	Вход Процедура Выход	Вход=поток_работ Процедура=алгоритм Выход=поток_работ
	Пускает поток лишь по одной исходящей ветви	Эксклюзивный шлюз	Вход Условие Выход	Вход=поток_работ Условие=алгоритм Выход=поток_работ
	После него могут идти лишь триггерные события	Шлюз, основанный на «Событиях»	Вход Событие Выход	Вход=поток_работ Событие=алгоритм Выход=поток_работ
	Активируется, только при наличии потока на каждой входящей ветви	Параллельный шлюз	Вход1 Вход2 Вход3 ВходN Выход	Вход1=поток_работ Вход2=поток_работ Вход3=поток_работ ВходN=поток_работ Выход=поток_работ
	Может соединяться с любым элементом потока с помощью ассоциации	Объект данных	Документ	Документ=текст
	Отношение между графическими объектами	Обычная связь	От К	От=графический_объект К=графический_объект
	Позволит отслеживать недопустимую последовательность элементов	Связь для шлюза, основанного на событиях	От К	От=графический_объект К=графический_объект
.....	Ассоциативное отношение между графическими объектами по ключу	Ассоциация	От К	От=графический_объект К=графический_объект

Таблица 28. Денотативная и сигнификативная семантика

визуального языка eEPC

Графичес-кое слово	Понятие	Класс	Свойства класса	Значения свойства класса
	Это факт свершения чего-либо, не имеющий продолжительности во времени, либо это время стремится к нулю (или не имеет значения)	Событие	Время	Число
	Исполнение функции всегда заканчивается событием	Функция	Функция	Функция=алгоритм
	Направление следования потоков работ	Путь процесса	От К	От=поток_работ К=поток_работ
	Объект отражает различные организационные звенья предприятия	Подразделе-ние	Название	Название=текст
	Логический оператор, определяющий связь между событиями и функциями в рамках процесса. Позволяет описывать ветвление процесса	Исключающее ИЛИ	От К	От=поток_работ К=поток_работ
	Логический оператор, определяющий связь между событиями и функциями в рамках процесса. Позволяет описывать ветвление процесса	ИЛИ	От К	От=поток_работ К=поток_работ
	Логический оператор, определяющий связь между событиями и функциями в рамках процесса. Позволяет описывать ветвление процесса	И	От К	От=поток_работ К=поток_работ
	Объект отражает носители информации	Информация (материал)	Документ	Документ=текст
	Основной бизнес-процесс	Главный процесс	От К	От=графический_объект К=графический_объект
	Структурно-функциональный элемент	Компонент	Наимено-вание	Наименование=текст
	Область исследования	Предметная область	Наимено-вание	Наименование=текст
	Набор простых процессов, собранных по одинаковому признаку и решающих одну и туже задачу	Группа процессов	Наимено-вание	Наименование=текст
	Отношение между графическими объектами	Динами-ческая соединитель-ная линия	От К	От=графический_объект К= графический_объект

8. Формализация ошибок денотативной и сигнификативной семантики диаграмматических моделей потоков проектных работ

К семантическим ошибкам диаграмматических моделей потоков работ относятся следующие ошибки [89-97].

Несоответствие синонимов (денотативная ошибка).

Темпоральные слова визуального языка $(\tilde{a}_l, t_i)$ и $(\tilde{a}_k, t_i)$ являются синонимами тогда и только тогда, когда $\tilde{a}_l \neq \tilde{a}_k, \tilde{a}_l \equiv \tilde{a}_k, t_i < t_j$ и обозначается синонимия слов как $(\tilde{a}_l, t_j) \equiv (\tilde{a}_k, t_j)$. Тождественное равенство слова определяет подобие (сходство) структуры и значений признаков денотата. Ошибкой является ситуация, когда наименование денотатов слов в двух темпоральных трассах графического языка схожие, но значения других признаков сильно разнятся. На практике такая ситуация представлена следующим образом: при анализе диаграмматической модели визуального языка выявляется структурное подобие слов и имен денотатов, но значения остальных признаков денотатов слов различны. В САПР возможно представлять варианты состава изделий при различных условиях: в исполнениях, заменяемости и взаимозаменяемости. В этой ситуации выполнение взаимозаменяемости таких компонентов изделия в виде графических слов в диаграмматической модели визуального языка является ошибкой несоответствия синонимов.

Несоответствие антонимов (денотативная ошибка).

Темпоральные слова визуального языка $(\tilde{a}_l, t_j)$ и $(\tilde{a}_k, t_j)$ являются антонимами тогда и только тогда, когда $\tilde{a}_l = \neg \tilde{a}_k, t_i < t_j$ и антонимия слов обозначается как $(\tilde{a}_l, t_l) \equiv (\neg \tilde{a}_k, t_k)$. Тождественная противоположность двух слов определяет подобие (сходство) структуры и противоположность (инверсность) значений признаков денотата. Как правило, слова «Начало»

и «Конец» в диаграмматике являются антонимами графического языка. Ошибкой является проектная ситуация, когда наименования денотатов слов в двух темпоральных трассах графического языка противоположные (инверсные), но значения других признаков очень схожие. На практике такая ситуация представлена следующим образом, при анализе диаграмматической модели визуального языка выявляется структурное тождество слов и инверсия имен денотатов, но значения остальных признаков денотата слова эквивалентны. В этой ситуации выполнение взаимозаменяемости таких слов в диаграмматической модели визуального языка является ошибкой несоответствия антонимов.

Конверсивность отношений заключается в связывании антонимов диаграмматических моделей визуальных языков, описывающих одну и ту же проектную ситуацию, но с разных ролей. Ошибка конверсивности отношений является сигнификативной, т. е. структурной (конструкционной), и определяется как отсутствие этих отношений между антонимами диаграмматических моделей, описывающих одну и ту же проектную ситуацию, но с разных ролей.

Несогласованность объектов является сигнификативной ошибкой. Заключается в отсутствии отношения между зависимыми темпоральными словами.

9. Метод преобразования проектных процессов

Динамическая реконфигурация бизнес-процессов требует наличия механизма преобразования диаграмм для достижения гибкости, улучшения функциональности и повышения эффективности существующего бизнес-процесса предприятия. В работах [36, 38, 39, 107] проблема реконфигурации была глубоко исследована и с теоретической, и с практической точки зрения. Авторами предлагается применять преобразование структуры диаграммы с помощью процедур удаления, вставки и замены с сохранением связности в течение (до и после) конкретного времени. Необходимо, чтобы все графические примитивы имели время-метку, по которой определяется время преобразования диаграммы. Как правило в BPMN, eEPC, IDEF0, UML AD и т. п. графические примитивы содержат описание (примечание в UML AD), которое можно определить, как переменную времени. Рассмотрим пример UML AD диаграммы (рис. 23).

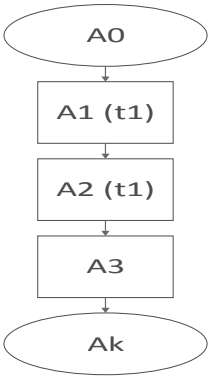


Рис. 23. UML AD диаграмма с меткого времени t_1

Графические примитивы (элементы) A_1 и A_2 имеют временные метки t_1. Это означает, что в определенный момент времени t_1 с данными элементами будут произведены определенные преобразования (операции): (1) Вставка, (2) Замена и (3) Удаление. Логично предположить, что в один промежуток времени над одним элементом может выполняться лишь одна операция. Поэтому для каждой

временной метки будет отведена лента, на которой для одного элемента будет возможность указать три варианта: 1 – Вставка, 2 – Замена, 3 – Удаление. Дополнительная информация при операции «Вставка/Замена» будет храниться в расширенной ленте, позволяющей хранить не просто числа, а множество квазитермов. Для операции 1 будем использовать дополнительную функцию insert(), позволяющую извлечь необходимую информацию из расширенной ленты и за счет подграмматики сформировать вставляемый фрагмент. Операция 2 является комплексной операцией и представляет собой совокупность операций удаления и вставки. Для этого вводится дополнительная функция replace(). На начальном этапе рассматривается удаление. В результате в момент времени t_1 диаграмма принимает вид, показанный на рис. 24.

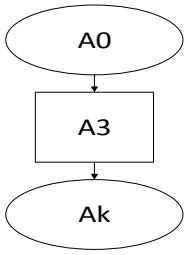


Рис. 24. Удаление элементов в диаграмме

Цепочка из удаляемых элементов может быть сколь угодно длинная. Для выполнения удаления будем использовать следующий способ. Если встречается элемент с отмеченной временной меткой, то ссылка на этот элемент заносится в магазин. Далее автомат следует по элементам, пока не встретит элемент с отсутствием временной метки. В таком случае выполняется специальная функция change_rel(), которая «достает» из магазина ссылку на первоначально удаляемый элемент и привязывает ее к текущему элементу. Процесс изображен на рис. 25.

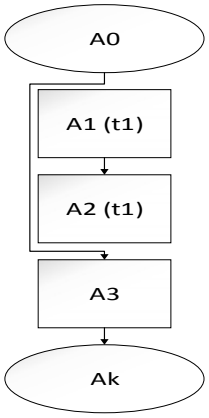


Рис. 25. Переназначение связей при удалении элемента

Для того чтобы не оставлять удаленные элементы висящими на диаграмме при прохождении удаляемых квазитермов, выполняется функция delete(), которая удаляет элемент из диаграммы. Отдельно стоит ввести функцию delete_with_link(). Она будет производить удаление элемента вместе с входящей связью. Грамматика для такой диаграммы представлена в таблице 29.

Таблица 29. Темпоральная RVT-грамматика для UML AD

Предыдущее состояние	**Квазитерм**	**След. состояние**	**Операция**
r_0	A0i	r_1	insert()/$W_3(k^{t(1)}==1)$
	A0	r_1	o
r_1	rel	r_2	o
r_2	Ai	r_1	insert()/$W_3(k^{t(1)}==1)$
	Ar	r_1	replace()/$W_3(k^{t(1)}==2)$
	Ad	r_3	(delete(), $W_1(l^{1m})$)/ $W_3(k^{t(1)}==3)$
	A	r_1	o
	Ak	r_5	o
r_3	drel	r_4	o
r_4	Ai	r_1	(change_rel(),insert())/$W_3(k^{t(1)}==1)$
	Ar	r_1	(change_rel(), replace()) /$W_3(k^{t(1)}==2)$
	Ad	r_3	delete_with_link()/$W_3(k^{t(1)}==3)$
	A	r_1	change_rel()
	Ak	r_5	change_rel()
r_5	no_label	r_k	*

Глава 2. Автоматизация процесса обучения проектировщика структурно-семантическому анализу и преобразованию потоков работ в автоматизации проектирования

1. Модели предметной области автоматизированного проектирования, квалиметрических характеристик проектировщика и сценарий обучения

Модель предметной области автоматизированного проектирования представлена в виде дерева онтологий и динамически использует иерархические, порядковые и ассоциативные связи онтологий объектов и процессов проектирования. Каждой онтологии соответствует учебный элемент. Иерархические связи используются для описания объекта и процесса проектирования с разной степенью детализации. Порядковые связи упорядочивают описание на одном иерархическом уровне и определяют цепочки онтологий. Ассоциативные связи соединяют иерархические и порядковые онтологии разных уровней. Модель диаграмматики (предметной области) позволяет адекватно представить учебный материал и является базой знаний промышленного проектирования. Модель предметной области имеет вид:

$$CADModel = \{ИмяОбъекта, Функции, Процессы, Данные, Паттерны, МетаДата| ortree, \prec, view\},$$

где $ИмяОбъекта=\{ИмяОбъекта_i, i=1…E\}$ – множество имен объектов проектирования;

$Функции=\{функция_i, i=1…Z\}$ – множество проектных функций;

$Процессы=\{процесс_i, i=1…P\}$ – множество проектных процессов;

$Данные=\{данные_i, i=1…D\}$ – множество проектных данных;

Паттерны={Операция$_i$, Команда$_i$, Способ$_i$, i=1…T} – множество проектных шаблонов,

Операция={операция$_i$, i=1…O} – множество проектных операций,

Команда={команда$_i$, i=1…C} – множество проектных команд,

Способ={способ$_i$, i=1…S} – множество проектных способов выполнения команды,

Атом={понятие$_i$, действие$_i$, i=1…A} – множество «атомов» знаний, состоящее из элементарных понятий и простейших действий,

Атом ∈ Этап, Атом ∈ Процедура, Атом ∈ Операция, Атом ∈ Команда, Атом ∈ Способ;

МетаДата={<ключ$_i$>, хэш-функция, i=1…H} – метаданные модели,

где <ключ$_i$> – кортеж ассоциативных ключей,

хэш-функция – хэш-функция поиска элемента;

ortree – иерархическое отношение;

$\prec$ – отношение порядка;

view – ассоциативное отношение.

Структура паттернов PatternOperation (проектных операций), PatternComand (проектных команд), PatternSposob (проектных способов) одинакова. Например, структура PatternOperation имеет вид: PatternOperation = {название, назначение, мотивация, применимость, структура, участники, отношения, результаты, реализация, пример, применения, родственные паттерны}.

Разработана модель обучаемого проектировщика, отражающая динамический уровень его подготовленности к решению проектных задач. Уровень описан нечеткими лингвистическими критериальными параметрами (знания, умения, навык и компетентность).

Описание модели обучающегося проектировщика имеет вид:

$$UserModel = \{OcenkaZnanie_i, OcenkaUmenie_i, OcenkaNavik_i,$$

$OcenkaKompetentnost_i$, haracteristika | calcZ, calcU, calcN, calcK, i=1…N},

где $OcenkaZnanie_i$, $OcenkaUmenie_i$, $OcenkaNavik_i$ и $OcenkaKompetentnost_i$ – массивы оценок знаний, умений, навыков и компетентности соответственно, N – число контрольных точек K_i сценария. Областью значений функций расчета указанных оценок являются пары (D, μ), calcZ, calcU, calcN, calcK $\in$(D, μ) где D – значение функции евклидово расстояние, μ – значение функции принадлежности к классу проектной характеристики [99], haracteristika={оценка$_1$, оценка$_2$, оценка$_3$, …, оценка$_S$} – множество лингвистических характеристик. calcZ:markTeor$_i$ $\rightarrow$ оценка$_i$, calcU: mark$_i$ $\rightarrow$ оценка$_i$, calcN:t$_i$ $\rightarrow$ оценка$_i$, calcK:calcZ, calcU, calcN $\rightarrow$ оценка$_i$, где markTeor$_i$ – множество оценок за решенные теоретические задачи, mark$_i$ – множество оценок за выполненные проектные операции в пакетах САПР, t$_i$ – множество из временных интервалов, затраченных на решение проектных задач [99, 100].

Реализация функций calcZ, calcU, calcN, calcK выполнена с помощью нечеткой карты Кохонена и разработанной оценочной шкалы (таблица 30) в процентном, интервальном и лингвистическом видах. Структурная схема нечеткой карты изображена на рис. 26.

Таблица 30. Оценочная шкала на множестве значений функции μ

$p_S = 100\times \mu_S$	Интервал расстояния (отклонения)	Лингвистическая характеристика
p_S	$[(1-\mu_{2S-1}) \times \sum_{e=1}^{card(X)} w_{ije}; (1-\mu_{2S}) \times \sum_{e=1}^{card(X)} w_{ije}]$	оценка$_S$

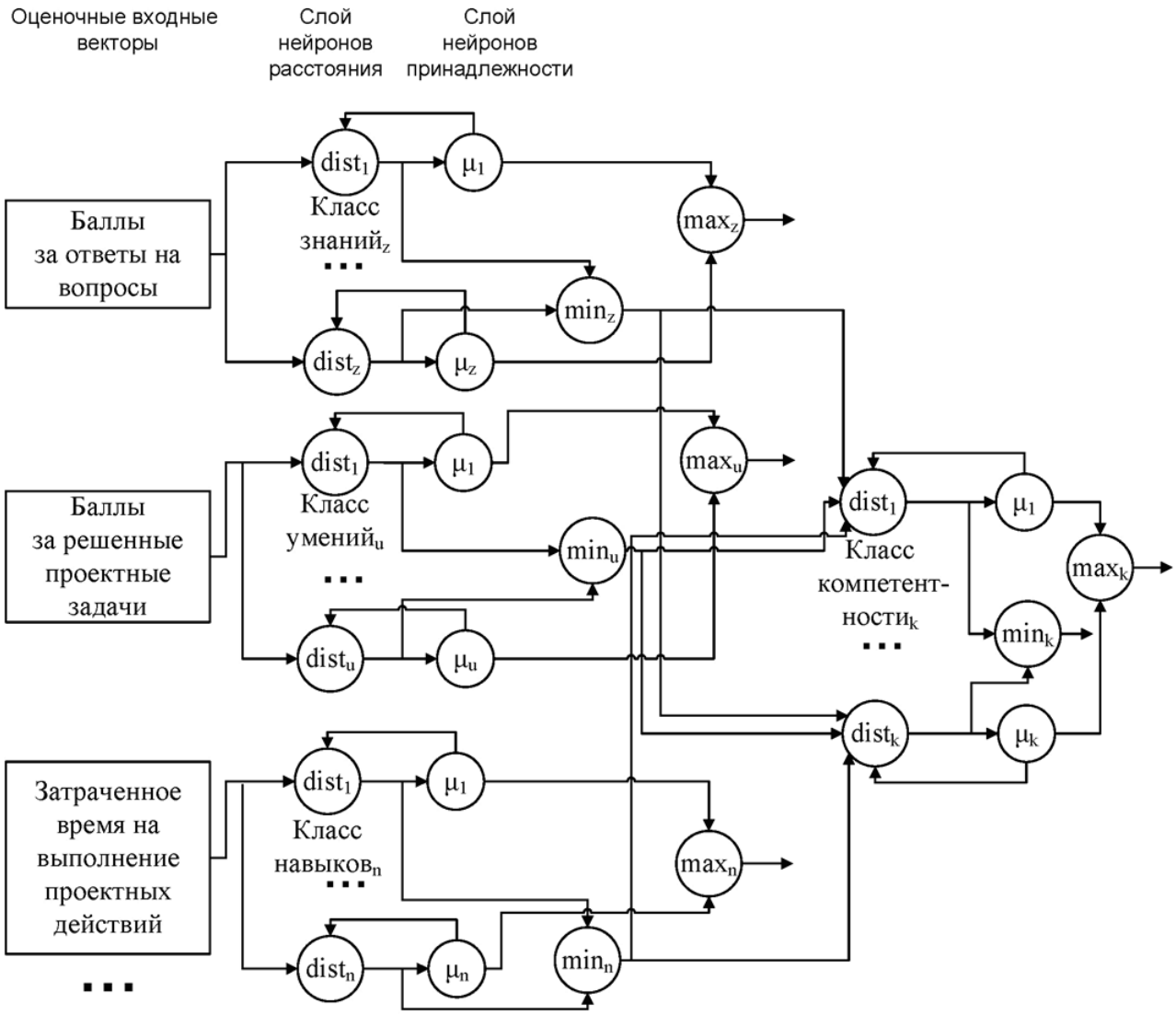


Рис. 26. Структурная схема карты Кохонена оценки проектных характеристик обучаемого проектировщика

Многоточие на рис. 26 означает возможность масштабируемости архитектуры нечетких карт. Функция принадлежности μ имеет вид:

$$\mu_{ij} = \exp\left(-\frac{1}{2} dist_{ij}^{\,2}\right), i = 1,...,C, j = 1,...,4,$$

где C – число подклассов класса; j имеет значения от 1 до 4 и означает четыре класса: знания, умения, навыки и компетентность.

Активность нейронов слоя расстояния вычисляется как Евклидово расстояние (отклонение) от требуемой активности

$$dist_{ij} = \sqrt{\sum_{e=1}^{T}(x_e - w_{ije})^2},$$

где T – число критериальных параметров входного оценочного вектора X, x_e – элемент вектора X, w_{ije} – вес дуги, связывающий элемент x_e и нейрон ij класса.

При многократном применении нечетких карт обратная связь позволяет обновлять весовые коэффициенты на каждом t шаге $W_{ij}(t+1)=W_{ij}(t)+ \mu(t)\times dist_{ij}(X, W_{ij}(t))$, что позволяет повысить точность оценки. Четыре выходных кортежа нечеткой карты имеют вид $<dist_j, \mu>$, j=1…4, где $dist_j$ – значение активности нейрона j класса, μ значение функции принадлежности входного вектора к конкретному j классу. Разработана модель сценария обучения. В основу модели сценария положена система, состоящая из ориентированного графа, отображений вершин и альтернативного выбора траектории обучения. Модель сценария имеет вид:

Scenariy = {G (vertex, edge), Reflaction, Alternativ},

где G (vertex, edge) – ориентированный граф сценария,

vertex = $\{v_i, i=1…V\}$ – множество атрибутивных вершин,

edge = $\{e_i, i=1…E\}$ – множество дуг;

Reflaction = $\{Rf_1, Rf_2, Rf_3, Rf_4\}$ – множество гетерогенных отображений вершин в объекты проектирования (Rf_1 – имена объектов проектирования, функции, процессы, данные, паттерны (см. модель CADModel), Rf_2 – тестовые вопросы, Rf_3 – практические проектные задачи, Rf_4 – контрольные точки K_i, содержащие требуемые (целевые) значения лингвистических критериальных параметров (проектных характеристик) обучаемого проектировщика);

Alternativ = $\{v_j$, если v_i инциндентна v_j и $v_i \neq i$, $v_i < v_j\}$ – выбор обучаемым проектировщиком траектории обучения из v_i неконтрольной вершины.

2. Метод адаптивного планирования и управления траекторией обучения

Метод адаптивного планирования и управления траекторией обучения содержит диагностику знаний, умений, навыков и компетентности обучаемого проектировщика. Диагностика основана на классификации с помощью нечетких карт Кохонена, которые хорошо себя зарекомендовали в качестве универсальных классификаторов.

Параметры модели обучаемого проектировщика меняются событийно в контрольных точках K_i сценария. Оценочные входные векторы поступают на вход нечеткой карты, которая классифицирует полученные данные и формирует нечеткие характеристики уровня подготовленности проектировщика. Число входов нейрона класса знаний равно числу элементов входного вектора баллов за ответы на вопросы. Число входов нейрона класса умений равно числу элементов входного вектора баллов за решенные проектные задачи. Число входов нейрона класса навыков равно числу элементов входного вектора затраченного времени на выполнение проектных действий. Число входов нейрона класса компетентности равно трем (знания, умения и навыки) (см. рис. 26). Классификация выполняется с помощью нечеткой карты и разработанной оценочной шкалы в процентном, интервальном и лингвистическом видах (см. табл. 30). В зависимости от уровня подготовленности обучаемого проектировщика принимается решение о выборе сценарной траектории обучения: продолжить обучение по статическому сценарию (траектории, определенной моделью сценария) или автоматически разработать динамический сценарий на основе моделей обучаемого и предметной области (проводится структурный синтез сценария). Реконструкция сценария изменяет предложенный статический сценарий, дополняя его сконструированным сценарием из элементов модели предметной области. Необходимость реконструкции сценария возникает тогда, когда

процедура диагностики (метод диагностики) показывает результат $\mu_т$, не удовлетворяющий заданным целевым проектным характеристикам обучаемого $\mu_ц$ в точке K_i: $\mu_т < \mu_ц$ (для знаний, умений, навыков и компетентности). Для упорядочивания этапов повышения уровней знаний, умений, навыков и компетентности проектировщика в обучении определена функция h, которая обладает минимальным значением принадлежности μ одному из классов знания, умения, навыка и компетентности: $h = \min(\mu_{знания}, \mu_{умения}, \mu_{навыки}, \mu_{компетентность})$. Применение функции h позволяет равномерно распределить весь учебный материал для получения сбалансированных проектных характеристик по всему курсу. Для учета в методе планирования и управления траекторией предыстории обучения проектировщика (его характеристик в модели обучаемого) все контрольные значения характеристик в контрольных точках K_i для OcenkaZnanie, OcenkaUmenie, OcenkaNavik, OcenkaKompetentnost определяются с помощью оператора И: $\wedge\mu_i$, i=1...N, где N – число контрольных точек K_i. Значения функции являются общим подмножеством всех функций μ_i на множестве значений.

Алгоритм автоматической разработки динамического сценария состоит в следующем.

1. Выбираются учебные элементы E из модели предметной области, связанные с контрольным элементом K_i.

2. Выбирается один критериальный параметр (знание, умение, навык или компетентность) обучаемого проектировщика P, имеющий минимальное значение μ.

3. Из множества E выбираются учебные элементы, которые связаны с P (образуя кортеж DE).

4. К кортежу DE добавляется из сценария пройденный контрольный теоретический и практический учебный материал, связанный с P.

5. Добавляется контрольный элемент K_i.

Обучаемый проектировщик проходит обучение по траектории, состоящей из элементов кортежа DE. Число выбранных учебных элементов может регулироваться наличием иерархической, порядковой и ассоциативной связанности элементов в модели предметной области.

3. Проект обучения

Предложенные модели и метод реализованы в интеллектуальной корпоративной среде обучения (ИКСо). Блок обучения построен по классической трехзвенной архитектуре: клиент, сервер приложений и сервер баз данных. В качестве платформы разработки использована технология Java Platform, Standard Edition, поддерживающая встроенные средства клиент-серверных приложений (технологии RMI и ISOAP). В качестве сервера баз данных используется MySQL. Система является масштабируемой, позволяет расширять функциональность с помощью добавления компонентов (плагинов). Плагины позволяют расширять как клиентскую, так и серверную части с помощью предоставления клиентского и серверного API. Для реализации клиент-серверной технологии на Java Platform, Standard Edition выбрана технология веб-сервисов. Протокол SOAP используется для обмена сообщениями между веб-службами. Блок рекомендаций построен на платформе .NET Framework. В качестве примера выбрана система автоматизированного проектирования KOMPAS-3D V16. Для обработки событий используется технология Automation, реализованная в виде библиотеки на языке программирования C#.

Проведен эксперимент по обучению среди студентов Ульяновского государственного технического университета численностью 16 человек. Для оценки начального уровня навыков был проведен предтест, по результатам которого сформированы две группы из 8 и 10 человек, со средним уровнем навыков 0,5. Группы обучались по материалам курса «Азбука КОМПАС». Первая группа проходила линейный сценарий обучения, вторая – адаптивный. После обучения стояла задача построить сборку крышки насоса, состоящей из 471 элемента. Результаты эксперимента приведены в таблице. При использовании адаптивного

сценария обучения средний уровень навыков студентов выше на 20% по сравнению с линейным за счет более детального обучения, что повлекло увеличение времени подготовки на 17%.

Таблица 31. Сравнение линейного и адаптивного сценариев обучения

	Средний уровень навыков до обучения	Среднее время обучения, ч	Средний уровень навыков после обучения	Среднее время построения сборки, ч	Общее затраченное время, ч
Линейный сценарий	0,5	1,65	0,745	2,8	4,5
Адаптивный сценарий	0,5	2	0,9	1,9	3,9
Выигрыш		**-17%**	**20%**	**50%**	**20%**

Система рекомендаций положительно сказывается на скорости выполнения проектных решений за счет уменьшения дублирующих операций, ошибочных и лишних действий. При этом уменьшается количество действий при выполнении проектной операции на 1, в среднем это дает прирост 25% на одну рекомендацию. Исключение дублирующих проектных операций уменьшает количество действий на 30% в зависимости от количества продублированных операций.

Выводы и рекомендации

1. Сформулированные на основе анализа **недостатки** грамматик, применяющихся для анализа и контроля диаграммных языков, являются **целевым мотивом** разработки такого класса методов обработки указанных языков, который обладал бы способностью анализа, контроля и перевода множества встречающихся в практике проектирования сложных технических систем диаграммных языков ППР, обеспечивал бы полноту контроля синтаксических и семантических ошибок, возможность их нейтрализации и формирования семантического значения диаграммных языков в терминах целевого языка перевода, а также улучшал характеристики по параметрам память-время.

2. Указанная цель достигнута. Разработан **новый** класс синтаксически-ориентированных графических грамматик (темпоральная автоматная графическая грамматика, автоматные графические грамматики, автоматные графические грамматики с нейтрализацией ошибок, иерархические графические грамматики, транслирующие графические грамматики, нечеткие графические грамматики), отличающехся от известных простотой синтеза, универсальностью, линейными временными характеристиками анализа, малыми затратами памяти, обеспечивающие полноту контроля, обнаружение синтаксических и семантических ошибок в диаграммах ППР, возможность продолжения анализа в случае наличия ошибок, получение семантического значения диаграмм потоков работ в терминах денотационной, операционной, денотативной и сигнификативной семантик, поддерживающих коллективное, в том числе и удаленное, проектирование потоков работ.

3. Разработанные методики синтеза и анализа нового класса грамматик обеспечивают корректность их построения.

4. Разработанный класс грамматик, а также сами грамматики языков UML, IDEF, eEPC, ПГС, сетей Петри, специализированного языка, BPMN рекомендуются для использования в качестве базового математического аппарата программных анализаторов диаграммных языков ППР, входящих в состав инструментальных средств проектирования сложных технических систем. Применение таких анализаторов уменьшает ошибки диаграммных моделей ППР.

5. Предлагаемая схема метатрансляции диаграммных языков ППР на базе RV-грамматик обеспечивает автоматизацию построения анализаторов на их основе и обработку появляющихся новых диаграммных языков ППР, что сокращает сроки их внедрения в практику проектирования.

6. Разработанные графические спецификации типов синтаксических и семантических ошибок для языков UML, IDEF, eEPC, BPMN, специализированнго языка рекомендуются для применения в практике проектирования сложных технических систем, что предотвращает появление ошибок в ситуативных и нормативных моделях ППР.

7. Формализованы денотативные и сигнификативные ошибки диаграмматических моделей потоков работ в автоматизации проектирования сложных технических систем: несоответствие синонимов, несоответствие антонимов, конверсивность отношений, несогласованность объектов.

8. Предложена онтология диаграмматических моделей потоков работ в автоматизации проектирования как система понятий графических слов диаграмматической модели.

9. Разработан метод преобразования диаграмматических моделей потоков работ в автоматизации проектирования, который реконструирует поток работ с целью повышения эффективности этого потока в плане

сокращения сроков выполнения проектных работ и повышения качества бизнес-процесса (системы потоков работ) предприятия.

10. Формализованы модели автоматизации обучения структурно-семантическому анализу, преобразованию потоков работ и методу планирования, управления процессом обучения.

11. Для предприятия рекомендуется:

- интегрировать предлагаемые программные средства анализа и контроля ППР в инструментальную среду проектирования сложных технических систем;
- с помощью программных средств провести анализ и контроль синтаксиса и семантики ППР в используемой диаграмматике;
- устранить синтаксические и семантические ошибки.

Библиографический список

1. Thiemich C., Puhlmann F. An agile BPM project methodology // Business Process Management. – Springer, Berlin, Heidelberg, 2013. – C. 291-306.
2. Seethamraju, R. & Seethamraju, J., 2009. Enterprise systems and Business Process Agility- A Case Study. In Proceedings of the 42nd Hawaii International Conference on System Sciences., pp.1-12.
3. Bider, I., Johannesson, P. & Perjons, E., 2010. In Search of the Holy Grail: Integrating social software with BPM.
   Experience report. In Enterprise, Business-Process and Information Systems Modeling, LNBIP, Vol. 50. Springer, pp.1-13.
4. Kindermann, H., 2013. Empowering process participants - the way to a truly agile business process management. [Online] Available at: http://www.onthemove-
   conferences.org/index.php/keynotes2013/2013keynotek
   indermann [Accessed 15 Augustus 2013].
5. Fowler, M. & Highsmith, J., 2001. The agile manifesto. Software Development, 9(8), pp.28-35.
6. Shore, J. & Warden, S., 2008. The art of agile. O'Reilly.
7. Agile Business Process Development: Why, How and When - Applying Nonaka's theory of knowledge transformation to business process development. https://www.researchgate.net/publication/266078141
8. Becker, J., Kugeler, M. & Rosemann, M., eds., 2011. Process Management: A Guide for the the Design of Business Processes. 2nd ed. Springer.
9. Sherehiy, B., W., K. & J.K., L., 2007. A review of enterprise agility: Concepts, frameworks, and attributes. International Journal of Industrial Ergonomics, 37, pp.445-60.

10. Highsmith, J., Orr, K. & Cockburn, A., 2000. E-Business Application Delivery, pp. 4-17. [Online] Available at: www.cutter.com/freestuff/ead0002.pdf.
11. A global Swiss company offering advanced intelligent application software for multiple business sectors. http://whitestein.com/
12. Gram Consulting, 2009. «Ba» for Management Development. [Online] Available at: http://gramconsulting.com/2009/04/ba-for-management-development/
13. Концептуальное моделирование компьютеризованных систем: учебное пособие / П.И. Соснин. – Ульяновск: УлГТУ, 2008. – 198 с.
14. Andersson, T., Andersson-Ceder, A. & Bider, I., 2002. State flow as a way of analyzing business processes-case studies. Logistics Information Management, 15(1), pp.34-45.
15. YAWL Foundation, 2004. YAWL. [Online] Available at: http://www.yawlfoundation.org/ [Accessed 05 February 2016].
16. Bider, I., 2014. Analysis of Agile Software Development from the Knowledge Transformation Perspective. In Johansson, B., ed. To appeare in 13th International Conference on Perspectives in Business Informatics Research (BIR 2014). Lund, Sweden. Springer, LNBIP.
17. IbisSoft, 2009. iPB Reference Manual. [Online] Available at: http://docs.ibissoft.se/node/3 [Accessed 05 February 2016].
18. Jalali, A., Wohed, P. & Ouyang, C., 2012. Aspect Oriented Business Process Modelling with Precedence. In Business Process Model and Notation, LNBIP, Vol. 125. Springer, pp.23-37.
19. Hasso Plattner Institut. http://bpt.hpi.uni-potsdam.de
20. Fu K. Structural methods of pattern recognition. – Moscow: Mir, 1977. – P.319.

21. Zhang D. Q., Zhang K. Reserved graph grammar: A specification tool for diagrammatic VPLs //Visual Languages. Proceedings. 1997 IEEE Symposium on. – IEEE. – pp. 284-291 (1997).
22. Costagliola G., Lucia A.D., Orece S., Tortora G. A parsing methodology for the implementation of visual systems. [Online] Available at: http://www.dmi.unisa.it/people/costagliola/www/home/papers/method.ps.gz [Accessed 05 February 2016].
23. Wittenburg K., Weitzman L. Relational grammars: Theory and practice in a visual language interface for process modeling (1996). [Online] Available at: http://citeseer.ist.psu.edu/wittenburg96relational.html [Accessed 05 February 2016].
24. Zhang K. B., Zhang K., Orgun M. A. Using Graph Grammar to Implement Global Layout for A Visual Programming Language Generation System. (2002).
25. Шаров, О. Г. Синтаксически-ориентированная реализация графических языков на основе автоматных графических грамматик / О. Г. Шаров, А.Н. Афанасьев // Программирование. – 2005. – №6. – С. 56–66.
26. Шаров, О. Г. Методы и средства трансляции графических диаграмм / О. Г. Шаров, А. Н. Афанасьев // Программирование. – 2011. – №3. – С. 65–76.
27. Ахо А. В., Сети Р. Ульман Дж.Д. Компиляторы: Принципы, технологии и инструменты. М. : Издательский дом «Вильямс», 2003.
28. Шаров, О. Г. Нейтрализация синтаксических ошибок в графических языках / О.Г. Шаров, А. Н. Афанасьев // Программирование. – 2008. – №1. – С. 61–66.

29. Шаров, О. Г. Методы и средства трансляции графических диаграмм / О. Г. Шаров, А. Н. Афанасьев // Программирование. – 2011. – Т. 37. – № 3. – С. 65–75. http://elibrary.ru/item.asp?id=16777695.
30. Roth C. Using Microsoft Visio 2010. – Pearson Education, 2011.
31. Paradigm V. Visual paradigm for uml //Hong Kong: Visual Paradigm International. Available at: http://www. visual-paradigm. com/product/vpuml/. Accessed April. – 2010. – Т. 15. – С. 2010.
32. Santos Jr P. S., Almeida J. P. A., Pianissolla T. L. Uncovering the organisational modelling and business process modelling languages in the ARIS method //International Journal of Business Process Integration and Management. – 2011. – Т. 5. – №. 2. – С. 130–143.
33. Hoffmann H. P. Deploying model-based systems engineering with IBM® rational® solutions for systems and software engineering //Digital Avionics Systems Conference (DASC), 2012 IEEE/AIAA 31st. – IEEE, 2012. – С. 1–8.
34. Yuan Wang, Yushun Fan Using (2004) Temporal Logics for Modeling and Analysis of Workflows. In: Proceedings of E-Commerce Technology for Dynamic E-Business. IEEE International Conference. DOI: 10.110 9/ CEC-EAST.2004.72.
35. Conrad Bock (2008) Introduction to business process and definition metamodel. U.S. Na-tional Institute of Standard and Technology. Manufacturing Engineering. https://www.nist.gov.
36. Alexander Afanasyev, Nikolay Voit, «Intelligent Agent System to Analysis Manufacturing Process Models», Proceedings of the First International Scientific Conference «Intelligent Information Technologies for Industry» (IITI'16) vol.451 of the series Advances in Intelligent Systems and Computing. Russia, pp. 395-403 (2016).
37. Alexander Afanasyev, Nikolay Voit, Rinat Gaynullin, «The Analysis of Diagrammatic Models of Workflows in Design of the Complex Automated

Systems», Proceedings of the First International Scientific Conference «Intelligent Information Technologies for Industry» (IITI'16) vol. 450 of the series Advances in Intelligent Systems and Computing. Russia, pp. 227–236 (2016).

38. A.N. Afanasyev, N.N. Voit, R.F. Gainullin, «Diagrammatic models processing in designing the complex automated systems», 10th IEEE International Conference on Application of Information and Communication Technologies (AICT). Baku, Azerbaijan, pp. 441–445 (2016).
39. A.N. Afanasyev, N.N. Voit, E.Yu. Voevodin, R.F. Gainullin, «Control of UML diagrams in designing automated systems software," Proceedings of the 9th IEEE International conference on Application of Information and Communication Technologies: AICT – 2015, pp. 285–288 (2015).
40. A.N. Afanasev, N.N. Voit, E.Yu. Voevodin, R.F. Gainullin, «Analysis of Diagrammatic Models in the Design of Automated Software Systems» Object Systems – 2015: Proceedings of X International Theoretical and Practical Conference (Rostov-on-Don, 10-12 May, 2015) / Edited by Pavel P. Oleynik. – Russia, Rostov-on-Don: SI (b) SRSPU (NPI), pp. 124–129 (2015).
41. Карпов, Ю. Г. MODEL CHECKING. Верификация параллельных и распределенных программных систем / Ю. Г. Карпов. – СПб.: БХВ-Петербург, 2010. – 560 с.
42. Верификация программы и темпоральные логики. URL: http://logic.pdmi.ras.ru/~yura/modern/034.pdf.
43. Калянов, Г. Н. Моделирование, анализ, реорганизация и оптимизация бизнес-процессов: учебное пособие / Г. Н. Калянов. – М.: Финансы и статистика, 2006. – 240 с. URL: http://www.twirpx.com/file/2204790/.

44. Saeedloei, Neda, and Gopal Gupta. «Timed definite clause omega-grammars»ю LIPIcs-Leibniz International Proceedings in Informatics. Vol. 7. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2010.
45. Kruchten, P. 1998. The Rational Unified Process. Addison-Wesley.
46. Booch, G.; Jacobson, I. & Rumbaugh, J. 1998. The Unified Modeling Language User Guide. Addison-Wesley.
47. Booch, G. 1994. Object-oriented Analysis and Design with Applications, 2nd edition. Addison-Wesley.
48. HOORA's homepage. URL: http://www.hoora.org/
49. Jacobson, I.; Christerson, M.; Jonsson, P., & Gunnar, O. 1992. Object-Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley.
50. Rumbaugh, J.; Blaha, M.; Premerlani, W.; Eddy, F. & Lorensen, W. 1991. Object-Oriented Modeling and Design. Prentice Hall.
51. Gilb, T. 2003. Competitive Engineering. Addison-Wesley.
52. Shlaer, S. & Mellor, S. 1988. Object-Oriented System Analysis: Modeling the World in Data. Yourdon Press Computing Series. Prentice-Hall.
53. Parnas, D.L. 1978. Software Requirements for the A-7 Aircraft. Technical Report 3876. Naval Research Laboratory. Washington DC.
54. Alford, M. 1977. A Requirements Engineering Methodology for Real Time Processing Requirements. IEEE Transaction on Software Engineering. Vol. 3. No.1. Pages 60−69.
55. Andriole, S. 1989. Storyboard Prototyping for Systems Design: A New Approach to User Requirements Analysis. Q E D Pub Co.
56. Marca, D.A. & McGowan, C.L. 1988. SADT: Structured Analysis and Design Techniques. McGraw-Hill.
57. Parviainen, Päivi, et al. «Requirements engineering inventory of technologies.» VTT PUBLICATIONS (2003).

58. YAHODA.URL: http://web.archive.org/web/20120220001353/http://anna.fi.muni.cz/yahoda/
59. CPN Tools. URL: http://cpntools.org/
60. Roméo. URL: http://romeo.rts-software.org
61. TimesTool. URL: http://www.timestool.com/documentation.shtml
62. Tina Toolbox. URL: http://projects.laas.fr/tina/
63. Visual Object Net++. URL: https: // www.techfak.uni-bielefeld.de/~mchen/BioPNML/Intro/VON.html.
64. Workflow Handbook 2005:/ Layna Fischer (editor), Published by Future Strategies Inc., Book Division.
65. A.N. Afanasev, N.N. Voit. Grammar-algebraic approach to the analysis and synthesis diagrammatically models of hybrid dynamic design workflows // Information-measuring and Control Systems, no. 12, pp. 69-78, 2017. URL: http://www.radiotec.ru/article/20138#english (cited by 25.05.2018).
66. Verification program and temporal logic. URL: http://logic.pdmi.ras.ru/~yura/modern/034.pdf (cited by 25.05.2018).
67. Арефьев, А. А. Язык граф-схем параллельных алгоритмов и его расширения / А. А. Арефьев, Ю. П. Кораблин, В. П. Кутепов // Программирование. – 1981. – №4. – С. 14–25.
68. Вельбицкий, И. В. Технологический комплекс производства программ на машинах ЕС ЭВМ и БЭСМ-6 / И. В. Вельбицкий, В. Н. Ходаковский, Л. И. Шолмов. – М. : Статистика, 1980. – 264 с.
69. Афанасьев, А. Н. Контроль информации в системах автоматизации проектирования / А. Н. Афанасьев, А. А. Гужавин, О. Г. Кокаев и др. – Саратов : СГУ, 1985. – 136 с.
70. Афанасьев, А. Н. Автоматная графическая грамматика / А. Н. Афанасьев, О. Г. Шаров // Вестник УлГТУ. – 2005. – №1. – С. 54-56.

71. Войт Н.Н., Уханова М.Е. Анализ потоков конструкторско-технологических работ // Системы проектирования, технологической подготовки производства и управления этапами жизненного цикла промышленного продукта (CAD/CAM/PDM – 2017) [Электронный ресурс] : тр. XVII междунар. науч.-практич. конфер., 12–14 декабря 2017 г, Москва / под общ. ред. А.В. Толока, Ин-т проблем упр. им. В.А. Трапезникова. – Электрон. текстовые дан. – М. : ИПУ РАН, 2017. – С. 373-377. URL: http://lab18.ipu.ru/projects/conf2017/data.pdf
72. Афанасьев, А. Н. Методы и средства обработки диаграммных языков / А. Н. Афанасьев // Сборник научных трудов Российской школы-семинара аспирантов, студентов и молодых ученых ИМАП-2009. – Ульяновск : УлГТУ, 2009. – С. 16-30.
73. Afanasev, A. N. Analysis of diagram languages of design processes representation in a CAD computerize system / A. N. Afanasev, V. V. Sychev // Proceedings of International Conference. Interactive Systems And Technologies: The Problem of Human-Computer Interaction. – Collection of scientific papers. – Ulyanovsk : 2009. – September. – P. 199-207.
74. Afanasev, A. Intelligent methods and tools for handling graphic workflows in computer-aided design of comlex systems / A. Afanasev // Proceedings of International Conference. Interactive Systems And Technologies: The Problem of Human-Computer Interaction. – Collection of scientific papers. – Ulyanovsk : ULSTU, 2011. – P. 95-100.
75. Afanasev, A. N. The concept of construction and realization of graphical editors for CAD / A. N. Afanasev, O. G. Sharov // Proceedings of the International Conference. Interactive Systems : The Problems of Human – Computer Interaction. – Ulyanovsk : UlSTU, 2003. – P. 248-252.
76. Глушков, В. М. Алгебра. Языки программирования / В. М. Глушков, Г. Е. Цейтлин, Е. Л. Ющенко. – Киев : Наукова думка, 1978. – 320 с.

77. Афанасьев, А. Н. Нейтрализация ошибок в диаграммных графических языках САПР / А. Н. Афанасьев, В. В. Сычев // Информационные технологии: межвузовский сб. науч. трудов. – Ульяновск : УлГТУ, 2008. – С. 187-192.
78. Афанасьев А.Н., Войт Н.Н. Грамматико-алгебраический подход к анализу и синтезу диаграмматических моделей гибридных динамических потоков проектных работ // Информационно-измерительные и управляющие системы. – 2017. - №12. -– С. 69-78. URL: http://www.radiotec.ru/article/20138
79. Costagliola, G. Positional grammars: a formalism for LR-like parsing of visual languages / G. Costagliola, A. D. Lucia, S. Orece, G. Tortora. – (http://www.dmi.unisa.it/people/costagliola/www/home/papers/tvl96.ps.gz).
80. Deufemia, V. A Grammar-based Approach to Specify and Implement Visual Languages / V. Deufemia // Master's thesis. – 2002. – (http://www.dia.unisa.it/dottorato/TESI/tesi-deufemia.pdf).
81. Степанов, П. А. Вычислительная модель визуального языка / П. А. Степанов, М. Ю. Охтилев // Изв. вузов. Приборостроение. – 2006. – №11. – С. 28-32.
82. Rekers, J., Schurr A. A parsing algorithm for context sensitive graph grammars / J. Rekers, A. Schurr // Tech. Rep. 95-05: Leiden University, Dept. of Computer Science. – the Netherlands, 1995. – (http://citeseer.ist.psu.edu/ rekers95parsing.html).
83. Zhang, D.-Q. A context-sensitive graph grammar formalism for the specification of visual languages / D.-Q. Zhang, K. Zhang, J. Cao // The Computer Journal. 2001. – Vol. 44. – №3. – Pp. 186-200. – (http://citeseer.ist.psu.edu/zhang01contextsensitive.html).

84. Афанасьев, А. Н. Процессоры обработки нечеткой информации / А. Н. Афанасьев, П. И. Соснин, О. Г. Кокаев. – Саратов : СГУ, 1988. – 124 с.
85. Ярушкина, Н. Г. Основы теории нечетких и гибридных систем / Н. Г. Ярушкина. – М. : Финансы и статистика, 2004. – 320 с.
86. Adamo, J.M. L. P. L. a fuzzy programming language: 1. Syn6tactic aspects / J. M. Adamo // Fuzzy Sets and Systems. – 1980. – v. 3. – №2. – P. 151–179.
87. Кириллов С.Ю., Войт Н.Н., Гордеев В.А. Разработка и исследование временной RV-грамматики для анализа и контроля структурных особенностей диаграмматических моделей динамических распределенных потоков работ // Сборник трудов IX Всероссийской школы-семинар «Информатика, моделирование, автоматизация проектирования» (ИМАП - 2017). – Ульяновск: УлГТУ, 2017. – С. 130-134. URL: http://iscad.ulstu.ru/sites/default/files/ИМАП%202017.pdf
88. Свидетельство № 2016616685 Российская Федерация. RV-анализатор диаграммного языка BPMN для MS Visio: свидетельство о государственной регистрации программы для ЭВМ / Афанасьев А. Н., Войт Н. Н., Кириллов С. Ю.; заявитель и правообладатель Ульян. гос. техн. ун-т. – № 2016616685; заявл. 19.04.2016; зарегистр. 16.06.2016.
89. Котцова, Е. Е. Лексическая семантика в системно-тематическом аспекте / Е. Е. Котцова. – Архангельск : Помор. гос. ун-т, 2002. – 203 с.
90. Кронгауз, М. А. Семантика: учебник для студ. лингв. фак. высш. учеб. заведений / М. А. Кронгауз. – 2-е изд., испр. и доп. // М. : Издательский центр «Академия. – 2005. – С. 171.
91. Кобозева, И. М. Лингвистическая семантика / И. М. Кобозева. – М: – URSS, 2004.

92. Klein M. Combining and relating ontologies: an analysis of problems and solutions //IJCAI-2001 Workshop on ontologies and information sharing. – 2001. – С. 53-62.
93. Волкова, Г. А. Создание «онтологии всего». Проблемы классификации и решения / Г. А. Волкова // Новые информационные технологии в автоматизированных системах. – 2013. – №. 16.
94. Митрофанова, О. А. Онтологии как системы хранения знаний / О. А. Митрофанова, Н. С. Константинова. СПб : – 2015.
95. Euzenat J. et al. Ontology matching. – Heidelberg : Springer, 2007. – Т. 18.
96. Мизогучи, Р. Шаг в направлении инженерии онтологий / Р. Мизогучи // Новости искусственного интеллекта. – 2000.– №1-2.– С.11-36.
97. Малиновкский, В.П. Использование онтологического подхода при моделировании жизненного цикла знаний в системе корпоративной памяти организации / В. П. Малиновкский // Новости искусственного интеллекта. – 2005. – №3.– С. 31-41.
98. Самуйлов, К. Е. Основы формальных методов описания бизнес-процессов: учеб. пособие / К. Е. Самуйлов, Н. В. Серебренникова, А. В. Чукарин, Н. В. Яркина. – М. : РУДН, 2008. – 130 с.
99. Войт, Н. Н. Разработка компонентной автоматизированной обучающей системы САПР на основе гибридной нейронной сети / Н. Н., Войт А. Н. Афанасьев // Автоматизация и современные технологии. – 2009. – № 3. – С. 14 – 18.
100. Войт, Н.Н. Организация когнитивной автоматизированной обучающей системы (КАОС) промышленных пакетов САПР / Н. Н. Войт, А. Н. Афанасьев // Обозрение прикладной и промышленной математики. – 2009. – Т. 16. – №1. – С. 405.

Научное издание

ВОЙТ Николай Николаевич
АФАНАСЬЕВ Александр Николаевич

СТРУКТУРНО-СЕМАНТИЧЕСКИЙ АНАЛИЗ ПОТОКОВ РАБОТ И ОБУЧЕНИЕ ПРОЕКТИРОВЩИКОВ В АВТОМАТИЗАЦИИ ПРОЕКТИРОВАНИЯ

Редактор О. Ф. Хисматуллина

ЛР № 020640 от 22.10.97.
Подписано в печать 06.11.2018. Формат 60×84/16.
Усл. печ. л. 8,37. Тираж 500 экз. (1-й з-д 1-50 экз.). Заказ 46.

Ульяновский государственный технический университет
432027, г. Ульяновск, ул. Северный Венец, д. 32.
ИПК «Венец» УлГТУ, 432027, г. Ульяновск, ул. Северный Венец, д. 32.

Дата подписания к использованию 06.11.2018.
ЭИ № 1246. Объем данных 1,5 Мб. Заказ № 46.

Ульяновский государственный технический университет
432027, Ульяновск, Сев. Венец, 32.
ИПК «Венец» УлГТУ, 432027, Ульяновск, Сев. Венец, 32.

Тел.: (8422) 778-113
E-mail: venec@ulstu.ru
venec.ulstu.ru