ПРОГРАММИРОВАНИЕ, 2022, № 3, с. 70–77

КОМПЬЮТЕРНАЯ ГРАФИКА И ВИЗУАЛИЗАЦИЯ

УДК 004.92

ОНТОЛОГИЧЕСКИ УПРАВЛЯЕМЫЕ СРЕДСТВА АВТОМАТИЗАЦИИ РАЗРАБОТКИ ПРИЛОЖЕНИЙ ВИЗУАЛЬНОЙ АНАЛИТИКИ

© 2022 г. С. И. Чуприна[a,*] (ORCID: 0000-0002-2103-3771), К. В. Рябинин[a,] (ORCID: 0000-0002-8353-7641), Д. В. Кознов[b,***] (ORCID: 0000-0003-2632-3193), К. А. Маткин[a,****] (ORCID: 0000-0002-0444-9476)**

[a] ФГАОУ ВО "Пермский государственный национальный исследовательский университет", 614068 Пермь, ул. Букирева, 15, Россия

[b] ФГБОУ ВО "Санкт-Петербургский государственный университет", 199034 Санкт-Петербург, Университетская наб., 7/9, Россия

**E-mail: chuprinas@inbox.ru*

***E-mail: kostya.ryabinin@gmail.com*

****E-mail: d.koznov@spbu.ru*

*****E-mail: matkin.k@yandex.ru*

Поступила в редакцию 20.12.2021 г.
После доработки 12.01.2022 г.
Принята к публикации 20.01.2022 г.

Статья посвящена вопросам автоматизации разработки управляемых онтологиями приложений визуальной аналитики. С этой целью предлагается использовать инструменты визуального проектирования приложений на базе диаграмм потоков данных, входящие в состав платформы SciVi. Указанные инструментальные средства, как и генерируемые с их помощью приложения визуальной аналитики, относятся к категории управляемых онтологиями систем. Предложена унифицированная модель прикладной онтологии для решения задач генерации исходного кода приложений на некотором целевом языке программирования с использованием расширяемого репозитория онтологических ресурсов SciVi, построенных в соответствии с предложенной моделью. При наличии в репозитории необходимых онтологий и качественном визуальном моделировании конвейера обработки данных встроенные инструменты ядра платформы, независимые от специфики решаемых задач, не только автоматизируют построение приложений визуальной аналитики посредством автоматической генерации кода, но и гарантируют его валидность.

DOI: 10.31857/S0132347422030037

1. ВВЕДЕНИЕ

В процессе проектирования и реализации приложений визуального анализа данных далеко не всегда удается найти инструментальное окружение, платформу, обеспечивающую автоматизацию всех этапов жизненного цикла разработки таких приложений для анализа данных в конкретных предметных областях. Во многом это объясняется трудоемкостью или ограниченными возможностями таких IDE по адаптации к специфике решаемой задачи и предпочтениям как разработчиков, так и конечных пользователей, не являющихся специалистами в области программирования.

Зачастую необходимо написание дополнительных скриптов для сбора и первичной обработки данных, а иногда и поддержка взаимодействия с интерфейсами прикладного программирования аналитических библиотек, например, для целей обучения моделей. Это требует привлечения к проекту высококвалифицированных разработчиков-программистов. Как следствие, не все эксперты оказываются в состоянии полноценно использовать современные достижения в области методов визуального анализа данных и самостоятельно настроить инструменты платформы под свои нужды из-за достаточно высокого порога вхождения. Снижение этого порога – важная задача современного витка цифровизации.

Практика показывает, что хороший баланс между гибкостью конфигурации и интуитивной понятностью интерфейса удается достичь, если декларировать аналитический процесс при помощи средств программирования потоков данных (англ. Data Flow Programming, DFP), а применение методов и средств онтологического инжиниринга в задачах интеллектуального анализа данных обеспечивает повышение семантической мощности аналитического процесса, увеличивая

тем самым скорость получения его результатов и их качество. Д. Доу (D. Dou) и др. в своем подробном исследовании практических подходов к онтологически управляемому анализу данных выделяют следующие роли онтологий в аналитическом процессе [1]:

1. Преодоление семантического разрыва между данными, приложениями, алгоритмами и результатами анализа путем внедрения в аналитические инструменты знаний о предметной области источника данных.

2. Привнесение априорных знаний в алгоритмы анализа, которые либо сопровождают аналитический процесс, либо позволяют сократить пространство информационного поиска в зависимости от конкретной решаемой задачи и преследуемой анализом цели.

3. Формальное описание конкретных алгоритмов анализа в виде последовательности шагов предварительной обработки, фильтрации, визуализации и интерпретации данных.

Указанные роли взаимосвязаны. Так, например, формализованные знания о предметной области анализируемых данных позволяют улучшить качество предварительной фильтрации, автоматизируя поиск и устранение избыточной или противоречивой информации [2, 3].

В рамках цикла исследований, проведенных в Пермском государственном национальном исследовательском университете за последнее десятилетие, предложен, формализован, реализован и протестирован на практике конкретный метод разработки онтологически управляемых программных систем для интеллектуального анализа данных. Полученные в ходе этих исследований результаты подтверждаются опытом других научных коллективов, занимающихся схожей проблематикой (например, [4–7]), и согласуются с мнением Д. Доу и др. относительно практической значимости онтологий в сфере интеллектуального анализа данных.

В работе изложены ключевые формальные положения и основные аспекты реализации предложенного метода, а также приводятся примеры его практического применения.

Визуальные языки DFP, подобно средствам визуального моделирования [8], обеспечивают одновременно и высокую вариативность описаний, и высокую наглядность, и интуитивность интерфейса. Подход к описанию процессов анализа данных посредством DFP применяется во многих популярных аналитических платформах, например, KNIME (`https://www.knime.com/`), Weka (`http://old-www.cms.waikato.ac.nz/ml/weka/`) или RapidMiner (`https://rapidminer.com/`). Программой на языке DFP является диаграмма потоков данных (англ. Data Flow Diagram, DFD) [9].

2. УНИФИЦИРОВАННАЯ МОДЕЛЬ ПРИКЛАДНОЙ ОНТОЛОГИИ В ЗАДАЧАХ ГЕНЕРАЦИИ ПРИЛОЖЕНИЙ ВИЗУАЛЬНОЙ АНАЛИТИКИ

Для целей генерации программного кода приложений визуальной аналитики реализованный в рамках разработанной авторами платформы SciVi (`https://scivi.tools/`) инструментарий визуального моделирования конвейера обработки потоков данных на основе DFD формально может быть описан в терминах теории множеств как $DFD = \{\Omega, \Lambda\}$, где $\Omega = \left\{\Theta_i \mid i = \overline{1,n}\right\}$ – множество экземпляров операторов преобразования данных, $\Lambda = \left\{\lambda_j \mid j = \overline{1,m}\right\}$ – множество связей по данным между экземплярами операторов, n – число шагов преобразования данных в DFD, m – число элементарных операций передачи данных в DFD.

Каждый Θ_i – это экземпляр некоторого оператора $\Delta^{(i)} \in \Phi$ из палитры операторов Φ, поддерживаемых платформой SciVi. Палитра Φ фактически выражает функциональность генерируемого приложения визуальной аналитики, так как представляет собой множество всех поддерживаемых в приложении действий по преобразованию данных (включая загрузку данных, фильтрацию, визуализацию и т.д.).

Оператор Δ можно формализовать следующим образом: $\Delta : \{I, S\} \to O$, где $I = \{I_k \mid k = \overline{1,|I|}\}$ – множество типизированных входов, $S = \{S_l \mid l = \overline{1,|S|}\}$ – множество типизированных параметров, $O = \{O_t \mid t = \overline{1,|O|}\}$ – множество типизированных выходов оператора.

Параметры отличаются от входов тем, что их значения для оператора задает пользователь, в то время как входы означиваются в соответствии со связями по данным. Таким образом, параметры можно иначе назвать “настройками” оператора.

Каждую связь λ_j можно детализировать как $\lambda_j = \{O_t^{(a)}, I_k^{(b)}\}$, где $O_t^{(a)}$ – выход экземпляра Θ_a оператора $\Delta^{(a)}$, данные из которого согласно DFD передаются в $I_k^{(b)}$ – вход экземпляра Θ_b оператора $\Delta^{(b)}$.

Будем использовать легковесные онтологии (англ. Lightweight Ontologies) с атрибутированными концептами-вершинами [10]: $O = \{T, R, A\}$, $A = \emptyset$, $T = \left\{T_p \mid p = \overline{1,|T|}\right\}$, $T_p = \left\{T_{name}, T_{def}, T_{attr}\right\}$, где R – множество связей между концептами, T_{name} – название концепта (как правило –“ строка), T_{def} – определение концепта (как правило, короткий текст), $T_{attr} = \left\{\langle K_q, X_q\rangle \mid q = \overline{1,|T_{attr}|}\right\}$ – набор атрибутов концепта в виде пар “ключ” (K_q) – “значение” (X_q). Ключи обычно представляются

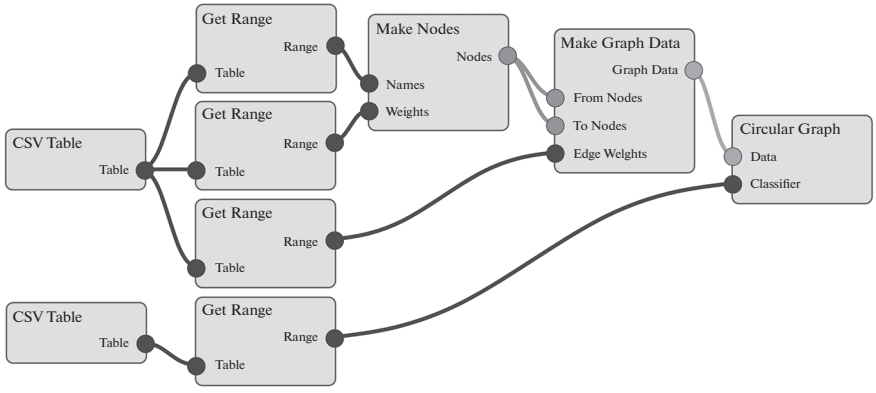

Рис. 1. DFD для загрузки, трансформации и визуализации данных в SciVi.

строками, а значения могут быть, вообще говоря, данными произвольного типа.

В основе предложенной нами унифицированной модели прикладной онтологии, управляющей процессом генерации кода приложений визуальной аналитики, выделяются две взаимосвязанных модели: модель предметных онтологий и модель онтологии задачи с визуальным представлением на базе DFD. Эти модели включают следующие базовые категории концептов:

- operator (оператор преобразования данных);
- input (вход оператора);
- output (выход оператора);
- setting (настроечный параметр оператора);
- type (тип данных);
- computing Resource (вычислительный ресурс);
- worker (экземпляр оператора, реализованный под определенный вычислительный ресурс);
- widget (элемент графического интерфейса);
- language (язык программирования, декларативный язык или язык разметки);
- dependency (вспомогательный модуль экземпляра оператора);
- network (вычислительная сеть);
- protocol (коммуникационный протокол);

Множество типов связей включает:

- is_a (“подкласс-класс”);
- a_part_of (“часть-целое”);
- has (“класс-свойство”);
- is_instance (“экземпляр-класс”);
- is_used (“использоваться для” или “использоваться в качестве” в зависимости от контекста);
- is_hosted (“исполняться на”);
- base_type (“быть базовым типом”);
- language (“быть написанным на языке”).

Каждый из базовых концептов входит в соответствующую таксономию понятий предметной области в качестве корневого элемента (связь “is_a” представляет таксономические отношения между дочерними и родительскими понятиями предметной онтологии). Связи других типов устанавливаются между дочерними концептами разных таксономий.

Редактор DFD в составе платформы SciVi реализован с использованием JavaScript на основе библиотеки Rete (`https://rete.js.org/`). Он обеспечивает пользователя высокоуровневым графическим интерфейсом для построения цепочек операторов преобразования и визуализации данных. Пример DFD, построенной в среде SciVi, показан на рис. 1.

3. ПРЕИМУЩЕСТВА УПРАВЛЯЕМЫХ ОНТОЛОГИЯМИ ПРОГРАММНЫХ СИСТЕМ

Основным преимуществом онтологически управляемых программных решений является их легкая расширяемость и адаптируемость к новым классам задач без внесения изменений в программный код системы. Однако надо предпринимать специальные усилия как для валидации программного кода таких систем в контексте их соответствия подходу ODSD (англ. Ontology-Driven Software Development – онтологически управляемая разработка ПО), так и валидации самих онтологий на предмет соответствия модели, управляющей работой программного средства. Для автоматизации этих процессов инструментальные средства в составе платформы SciVi также реализованы на ODSD-принципах.

При нашем подходе к разработке средств визуальной аналитики адаптируемость выражается в возможности унифицированного пополнения репозитория системы новыми компонентами визуализации, семантическими фильтрами, средствами интерактивного взаимодействия с графическими объектами и др. компонентами, онтологическое описание которых выполнено на базе единой онтологической модели, описанной в разделе 2. Это позволяет непосредственно сразу после пополне-

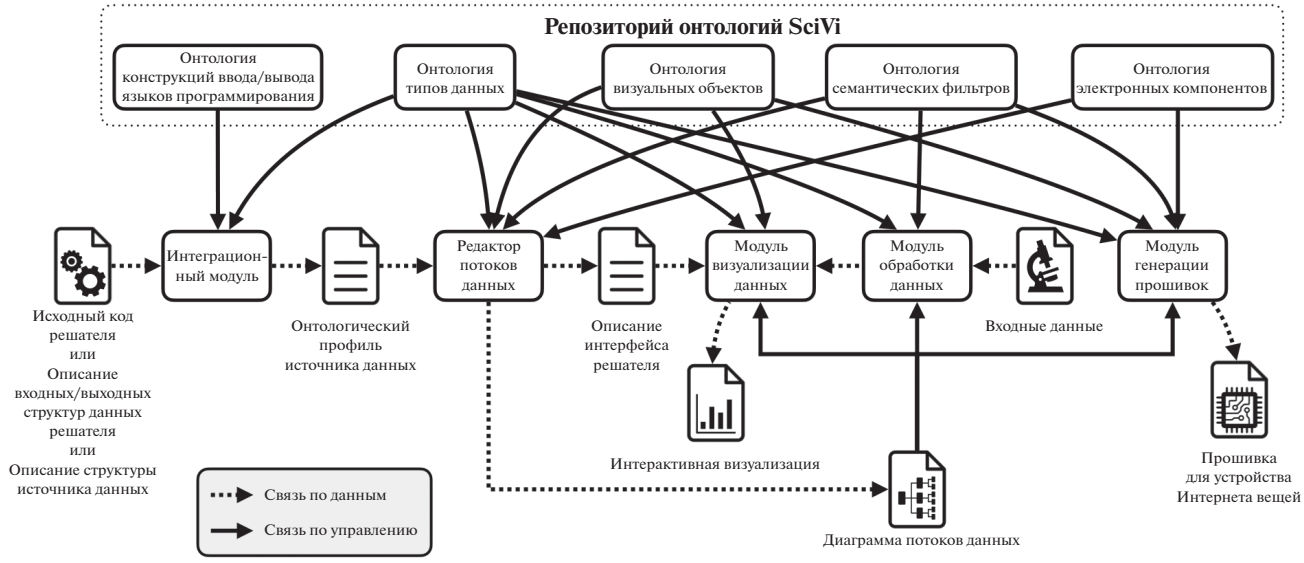

Рис. 2. Обобщенная архитектура онтологически управляемой платформы визуальной аналитики SciVi.

ния онтологической базы знаний без внесения изменений в программный код самих инструментальных средств использовать новые компоненты в процессе визуального проектирования аналитических приложений на базе DFD.

Интерактивность играет ключевую роль в процессе визуального анализа сложноструктурированных данных и обычно реализуется в соответствии с “Мантрой Шнайдермана” – “Сначала обзор, затем масштабирование и фильтрация, затем детализация по запросу” (англ. Schneiderman’s Mantra “Overview First, Zoom and Filter, then Details on Demand”) [11]. Современные интерактивные средства должны обеспечивать возможность пользователю непосредственно в процессе визуального анализа масштабировать поле просмотра изображения, выполнять разнообразную фильтрацию отображаемых данных, детализировать/обобщать визуальное представление заданной выборки данных и т.п.

Практически любое современное ПО для визуальной аналитики в той или иной степени поддерживает интерактивные возможности, сформулированные в “Мантре Шнайдермана”. Однако ПО, построенное на традиционных принципах разработки, не позволяет добавлять новые типы интерактивных взаимодействий в динамике, непосредственно во время анализа: для реализации новой функциональности требуется, как минимум, доступ к исходному коду и, возможно, нетривиальная настройка рабочего места для осуществления компиляции и компоновки. Онтологически управляемые программные продукты позволяют осуществлять изменение своего пользовательского интерфейса и внутренней функциональности через модификацию онтологии средствами высокоуровневого графического интерфейса. Таким образом, появляется возможность внесения корректив, требуемых для обеспечения необходимой эргономики при решении конкретной аналитической задачи без привлечения высококвалифицированных программистов.

Концепция автоматизированной разработки управляемых онтологиями систем визуальной аналитики реализована на практике с использованием инструментальных средств платформы SciVi [12–16], архитектура которой в обобщенном виде представлена на рис. 2.

4. УРОВНИ АДАПТИРУЕМОСТИ К АНАЛИТИЧЕСКИМ ЗАДАЧАМ

Управляемая онтологиями платформа визуальной аналитики SciVi предполагает 4 уровня адаптируемости к решаемым задачам:

1. Системный уровень: системные программисты адаптируют SciVi к решению принципиально новых задач, отличных от интеллектуального анализа данных. Так как данная работа посвящена автоматизации построения только приложений визуальной аналитики, то мы не будем останавливаться на системном уровне адаптации.

2. Прикладной уровень: прикладные программисты адаптируют SciVi к новым классам задач визуальной аналитики, унифицированным образом пополняя репозиторий системы новыми плагинами при помощи высокоуровневого интерфейса. Как правило, с технической точки зрения плагин представляет собой сравнительно небольшую программную библиотеку (или скрипт), реализующую отдельную функцию обработки/визуализации данных. Плагины могут создаваться как

с нуля, так и на основе готовых сторонних библиотек (являясь, фактически, легковесными обертками для этих библиотек). Разработка новых плагинов – основной путь расширения функциональности платформы SciVi на прикладном уровне.

3. Уровень базы знаний: разработчик с правами администратора базы знаний адаптирует SciVi к новым предметным областям посредством создания новой, либо модификации уже существующей управляющей предметной онтологии. Изменение базы знаний требуется в 3 случаях:

3.1. Когда прикладным программистом разрабатывается новый плагин, и его требуется внедрить в платформу. В этом случае чаще всего прикладной программист сам выступает в роли администратора базы знаний, внося в нее сведения о новом плагине.

3.2. Когда необходимо выбрать из всего множества существующих плагинов некоторое подмножество, предназначенное для решения определенного круга задач. Это действие может быть выполнено администратором базы знаний, не имеющим квалификации программиста, но обладающим навыками построения онтологий. Администратор базы знаний может адаптировать инструментальное окружение платформы под решение определенной задачи, включив в палитру инструментов для создания DFD только релевантные этим задачам средства, что упрощает процесс визуального проектирования конвейера обработки и анализа данных.

3.3. Когда требуется в каких-то пределах изменить функциональность плагинов и/или внести изменения на уровне интерфейса работы с DFD. Например, простым изменением онтологического описания можно модифицировать набор доступных на уровне интерфейса настроек, входов и выходов оператора (плагина), превратить ряд настроек во входы (и наоборот, при необходимости) и т.п. При этом не требуется переписывать исходный код плагина, а, следовательно, не нужно привлекать для таких модификаций прикладного программиста.

4. Внешний уровень редактора DFD: пользователи-аналитики адаптируют SciVi к конкретной аналитической задаче в рамках некоторой предметной области. На этом уровне, с использованием описанных предметных онтологий операторов (плагинов), строится DFD для решения конкретной аналитической задачи. Построение DFD осуществляется в среде высокоуровневого графического редактора, что не требует от пользователя ни квалификации инженера по знаниям, ни квалификации программиста. Таким образом, на этом уровне с платформой SciVi могут работать не только перечисленные выше категории разработчиков, но и эксперты в конкретной предметной области.

Предметные онтологии, создаваемые инженером по знаниям (с правами администратора базы знаний SciVi), объединены в репозиторий, и пользователь (аналитик) может выбирать, на основе каких онтологий инструментальные средства SciVi должны генерировать программный код конкретного аналитического приложения.

5. ОНТОЛОГИЧЕСКИ УПРАВЛЯЕМАЯ ГЕНЕРАЦИЯ ПРИЛОЖЕНИЙ ВИЗУАЛЬНОЙ АНАЛИТИКИ

Согласно описанной выше концепции, платформа визуальной аналитики (см. рис. 2) включает в себя репозиторий онтологий $\mathcal{D}=\{T, D_i \mid i=\overline{1,n}\}$, где T – онтология типов данных, n – общее число поддерживаемых платформой операторов загрузки, трансформации, фильтрации, визуализации и анализа данных, а D_i – онтология, описывающая оператор Δ_i. Пополняя этот репозиторий в ходе решения практических задач научной визуализации и визуальной аналитики, мы приняли решение хранить онтологические описания типов и операторов по отдельности, а не в виде одной общей онтологии. Это оказалось более удобным для инженеров по знаниям (выполняют функции администраторов онтологически управляемой платформы), так как создавать и модифицировать небольшие онтологии намного проще. В настоящее время ведутся работы по созданию высокоуровневого интерфейса к репозиторию онтологий SciVi со встроенными средствами семантического поиска “по образцу”. Такой интеллектуальный репозиторий онтологических ресурсов в составе SciVi позволит еще в большей степени автоматизировать разработку аналитических приложений.

Каждая онтология D_i описывает не только типизированные входы, настроечные параметры и выходы оператора Δ_i, но и множество его реализаций для различного вычислительного оборудования. Реализации описываются ссылками на динамические библиотеки или интерпретируемые скрипты. Так, например, операторы, выполняющиеся на стороне тонкого клиента, реализованы в виде скриптов на языке JavaScript, а операторы, выполняющиеся на стороне сервера, – в виде скриптов на языке Python, либо соответствующих скомпилированных исполняемых файлов. Реализации операторов могут быть как полностью автономными, так и зависящими от сторонних библиотек. В этом случае репозиторий системы пополняется необходимыми библиотеками функций.

Онтология T описывает поддерживаемые платформой визуальной аналитики типы данных, а также их соответствия конкретным типам данных в различных языках программирования. Это позволяет автоматизировать труд разработчика путем генерации интерфейса оператора и последу-

ющей проверки согласованности реализации с соответствующим онтологическим описанием. Например, для оператора, возвращающего максимальное и минимальное значения из таблицы числовых данных, генерируется следующий код в формате TypeScript Definition для проверки типов входных и выходных параметров:

```
declare type Inputs = {
  readonly ["Table"]: Array<Array<number>;
};
declare type Outputs = {
  ["Min"]: number;
  ["Max"]: number;
};
```

При запуске платформы программный менеджер репозитория предлагает пользователю задать предметную область Ф посредством выбора одной или нескольких онтологий, то есть осуществить выборку подмножества $\mathfrak{D}' = \{T, s(D_i) \mid i = \overline{1,n}\}$, где правило выбора s задается как

$$s(D_i) = \begin{cases} D_i, & \Delta_i \in \Phi, \\ \varnothing, & \Delta_i \notin \Phi. \end{cases}$$

Затем автоматически выполняется процесс объединения выбранных онтологий и построения общей управляющей онтологии $D = merge(T, D_1, \ldots, D_n)$. Здесь *merge* – функция объединения онтологий по следующим правилам:

1. Если имена вершин V_1 и V_2 из разных онтологий совпадают, то отдельно рассматривается случай, когда эти вершины представляют входы, настроечные параметры или выходы:

1.1. Если справедливы утверждения дескрипционной логики $\mathcal{ALC}$ [17] $V_1 \equiv Input \sqcup Setting \sqcup \sqcup Output$, $V_2 \equiv Input \sqcup Setting \sqcup Output$, то V_1 и V_2 входят в онтологию D как отдельные вершины.

1.2. Иначе V_1 и V_2 входят в онтологию D как одна вершина V. Множество инцидентных ей дуг $R(V) = R(V_1) \cup R(V_2)$, множество ее атрибутов $A(V) = A(V_1) \cup A(V_2)$. Атрибуты представляются в формате “ключ-значение”. Это обеспечивает отсутствия дублирований в описании типов данных и др.

2. Если для случая (1.2) выявлен конфликт атрибутов (различные значения атрибутов для одного и того же ключа), то генерируется предупреждение (используется лог системы), а значение атрибута берется из $A(V_1)$.

Таким образом построенная онтология D активно используется в процессе визуального проектирования конвейера обработки данных на основе DFD. В процессе генерации кода приложения встроенный в инструментальное окружение SciVi механизм логического вывода выполняет интерпретацию онтологии D и компонует приложение визуальной аналитики из реализаций операторов по аналогии со сборкой приложения микросервисной архитектуры. Реализация каждого оператора в сборке выступает в роли отдельного микросервиса, а сама платформа – в роли их общей шины. Для каждого оператора выполняется генерация кода:

1. Программного модуля исполнения оператора через запуск необходимого интерпретатора либо передачу управления соответствующим функциям динамической библиотеки. В обоих случаях обеспечивается необходимый уровень изоляции исполнения оператора от платформы.

2. Программного интерфейса бесшовной и прозрачной для пользователя связи с другими операторами, где передача данных внутри одного вычислительного узла осуществляется либо с использованием общей памяти, либо посредством автоматической сериализации и маршалинга данных между вычислительными узлами [12].

3. Графического интерфейса пользователя, включающего в себя:

3.1. Отображение соответствующей оператору вершины DFD (содержащей описанные для оператора входы и выходы).

3.2. Набор виджетов для управления значениями настроечных параметров оператора.

Сгенерированное таким образом приложение становится доступно пользователю через графический Web-интерфейс.

6. ПРИМЕРЫ ПРАКТИЧЕСКОГО ПРИМЕНЕНИЯ ПРЕДЛОЖЕННОГО ПОДХОДА

Разработанная платформа визуальной аналитики использована на практике для решения широкого круга задач из различных предметных областей (см. работы [13] и библиографию к ним). Приведем некоторые примеры, демонстрация и обсуждение которых состоялись в рамках 31-й Международной конференции по компьютерной графике и машинному зрению ГрафиКон 2021 [15, 16]. Примеры управляющих онтологий приведены в публикациях [12, 13].

Первый пример относится к области визуального анализа данных, генерируемых нейроинтерфейсами. На рис. 3 продемонстрирована визуализация результатов работы алгоритма классификации на основе общих пространственных шаблонов (англ. Common Spatial Pattern, CSP) данных электроэнцефалограммы (ЭЭГ) информанта, которому во время эксперимента на экране предъявлялись слова с разными лингвистическими характеристиками (например, переходные/непереходные глаголы и т.п.) [15]. Такая визуализация позволяет проанализировать, активность каких зон мозга

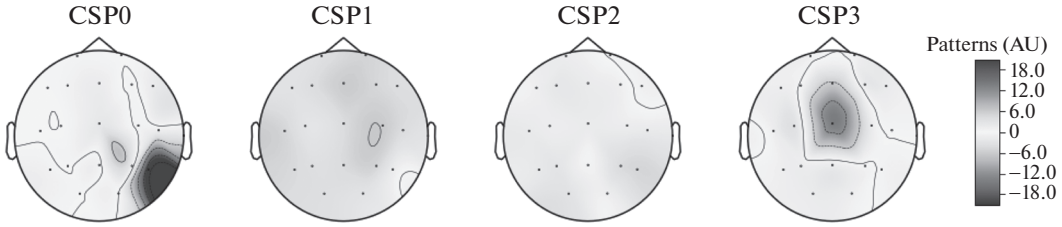

Рис. 3. Визуализация результатов работы алгоритма CSP на данных ЭЭГ.

в наибольшей степени различается при прочтении информантом соответствующих слов и, как следствие, выявить связь физиологических реакций человека на языковые особенности предъявляемого текстового материала. В данном примере платформа визуальной аналитики отвечает не только за построение графического представления данных, но и за проведение всего эксперимента в целом. Операторы визуализации выполняются на стороне тонкого клиента, а операторы демонстрации стимулов и сбора данных с электроэнцефалографа – на стороне сервера.

Второй пример посвящен генерации средств визуального анализа в задачах компьютерной лингвистики, где необходима визуализация фиксаций взгляда информанта, читающего текст с плаката, который размещен на сцене в виртуальной реальности [16] (рис. 4). Диаметр кругов пропорционален длительности соответствующих фиксаций. В данном случае платформа визуальной аналитики также управляет всем экспериментом целиком: операторы, работающие в браузере эксперта-аналитика, обеспечивают одновременно и управление виртуальной сценой, и сбор данных о направлении взгляда информанта, и последующий визуальный анализ этих данных. Визуализация позволяет сравнить полученные траектории движений глаз с известными шаблонами и выявить особенности восприятия информации в виртуальной иммерсионной среде.

7. ЗАКЛЮЧЕНИЕ

Применение методов и средств онтологического инжиниринга позволяет повысить уровень автоматизации разработки приложений визуальной аналитики, гибкость и настраиваемость ПО на специфику проблемной области решаемой задачи и индивидуальные предпочтения пользователя с точки зрения выбора инструментов анализа данных. Расширение функциональности онтологически управляемой системы визуальной аналитики оказывается возможным путем пополнения управляющих онтологий этой системы без переписывания ее исходного кода. В статье представлена унифицированная модель прикладной онтологии, управляющая работой инструментальных средств платформы SciVi, предназначенных для генерации программного кода приложений визуальной аналитики.

Онтологически управляемая платформа визуальной аналитики имеет характер динамического генератора приложений под конкретные нужды пользователей, и каждый получаемый пользователем экземпляр аналитического приложения содержит в себе лишь нужные пользователю функции. Вопросы валидации онтологически управляемых приложений, созданных на базе платформы SciVi, но развивающихся вне ее среды за счет расширения своей онтологической базы знаний, выходят за рамки данной работы. На практике обычно после внесения изменений в онтологию уже сгенерированного ранее приложения выполняется повторная генерация кода, что снимает проблемы его валидации при условии адекватно построенной DFD.

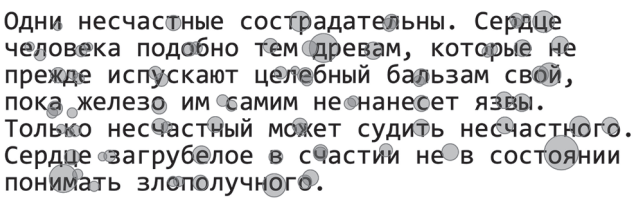

Рис. 4. Визуализация фиксаций взгляда в процессе чтения текста.

СПИСОК ЛИТЕРАТУРЫ

1. *Dou D., Wang H., Liu H.* Semantic Data Mining: A Survey of Ontology-Based Approaches // Proceedings of the 2015 IEEE 9th International Conference on Semantic Computing (IEEE ICSC 2015). 2015. P. 244–251. https://doi.org/10.1109/ICOSC.2015.7050814.
2. *Khasawneh N., Chan C.-C.* Active User-Based and Ontology-Based Web Log Data Preprocessing for Web Usage Mining // 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006 Main Conference Proceedings) (WI'06). 2006. P. 325–328. https://doi.org/10.1109/WI.2006.32.
3. *Perez-Rey D., Anguita A., Crespo J.* OntoDataClean: Ontology-Based Integration and Preprocessing of Distributed Data // ISBMDA 2006: Biological and Medi-

cal Data Analysis. 2006. P. 262–272.
https://doi.org/10.1007/11946465_24

4. *Li Y., Thomas M.A., Osei-Bryson K.-M.* Ontology-Based Data Mining Model Management for Self-Service Knowledge Discovery // Information Systems Frontiers. 2017. V. 19. P. 925–943.
https://doi.org/10.1007/s10796-016-9637-y
5. *Joshi K., Verma A., Kandpal A., Garg S., Chauhan R., Goudar R.H.* Ontology Based Fuzzy Classification of Web Documents for Semantic Information Retrieval // 2013 Sixth International Conference on Contemporary Computing (IC3). 2013. P. 1–5.
https://doi.org/10.1109/IC3.2013.6612160.
6. *Niskanen I., Kantorovitch J.* Ontology Driven Data Mining and Information Visualization for the Networked Home // 2010 Fourth International Conference on Research Challenges in Information Science (RCIS). 2010. P. 147–156.
https://doi.org/10.1109/RCIS.2010.5507374.
7. *Garanina N., Anureev I., Sidorova E., Koznov D., Zyubin V., Gorlatch S.* An Ontology-Based Approach to Support Formal Verification of Concurrent Systems // Formal Methods. FM 2019 International Workshops. 2020. P. 114–130.
https://doi.org/10.1007/978-3-030-54994-7_9
8. *Кознов Д.В.* Основы визуального моделирования. М.: Интернет-Университет Информационных Технологий; БИНОМ. Лаборатория знаний, 2007. 248 с.
9. *Sousa T.B.* Dataflow Programming Concept, Languages and Applications // Doctoral Symposium on Informatics Engineering. 2012. V. 130.
10. *Golitsyna O.L., Maksimov N.V., Okropishina O.V., Strogonov V.I.* The Ontological Approach to the Identification of Information in Tasks of Document Retrieval // Automatic Documentation and Mathematical Linguistics. 2021. V. 46. P. 125–132.
https://doi.org/10.3103/S0005105512030028
11. *Shneiderman B.* The Eyes Have It: a Task by Data Type Taxonomy for Information Visualizations // Proceedings 1996 IEEE Symposium on Visual Languages. 1996.
https://doi.org/10.1109/VL.1996.545307.
12. *Ryabinin K., Chuprina S., Labutin I.* Tackling IoT Interoperability Problems with Ontology-Driven Smart Approach // Lecture Notes in Networks and Systems. Springer, 2021. V. 342. P. 77–91.
https://doi.org/10.1007/978-3-030-89477-1_9
13. *Ryabinin K.V., Belousov K.I., Chuprina S.I., Shchebetenko S.A., Permyakov S.S.* Visual Analytics Tools for Systematic Exploration of Multi-Parameter Data of Social Web-Based Service Users // Scientific Visualization. 2018. V. 10. № 4. P. 82–99.
https://doi.org/10.26583/sv.10.4.07
14. *Ryabinin K.V., Belousov K.I.* Visual Analytics of Gaze Tracks in Virtual Reality Environment // Scientific Visualization. 2021. V. 13. № 2. P. 50–66.
https://doi.org/10.26583/sv.13.2.04
15. *Ryabinin K., Belousov K., Chumakov R.* Visual Analytics Tools for Polycode Stimuli Eye Gaze Tracking in Virtual Reality // Proceedings of the 31st International Conference on Computer Graphics and Vision (GraphiCon 2021). 2021. P. 211–222.
https://doi.org/10.20948/graphicon-2021-3027-211-222.
16. *Ryabinin K., Chuprina S., Labutin I.* Ontology-Driven Toolset for Audio-Visual Stimuli Representation in EEG-Based BCI Research // Proceedings of the 31st International Conference on Computer Graphics and Vision (GraphiCon 2021). 2021. P. 223–234.
https://doi.org/10.20948/graphicon-2021-3027-223-234.
17. *Baader F., Calvanese D., Mcguinness D., Patel-Schneider P., Nardi D.* The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, 2003. 555 p.