ИНФОРМАТИКА
ИНФОРМАЦИОННЫЕ СИСТЕМЫ

УДК 004.42

Семантические фильтры входных данных для системы научной визуализации SciVi

А. Ю. Бортников
Пермский государственный национальный исследовательский университет
Россия, 614990, Пермь, ул. Букирева, 15
a.yu.bortnikov@yandex.ru; +7-912-49-241-09

Рассматривается метод для расширения системы научной визуализации SciVi, предоставляющий дополнительные возможности более тонкой настройки на специфику решаемых в этой системе прикладных задач. Предлагается метод построения семантических фильтров средствами графического интерфейса пользователя. Расширяемость и гибкость предложенного решения достигается за счет управления работой программы посредством онтологической базы знаний о фильтрах. Для осуществления настройки системы визуализации на специфику конкретной задачи пользователю достаточно провести связи между узлами, описывающими входные и выходные данные решателя, и выбранными для визуализации объектами. В качестве промежуточных узлов полученной в результате этого диаграммы потока данных выступают семантические фильтры, предлагаемые пользователю на выбор из репозитория системы. Клиент-серверная архитектура приложения позволяет распределить нагрузку в соответствии с потребностями пользователей.

Ключевые слова: *научная визуализация; поток данных; семантический фильтр; онтологический инжиниринг; фильтрация данных.*

DOI: 10.17072/1993-0550-2016-4-37-42

Введение

Высокое разнообразие и большой объем данных является одной из основных проблем в научной визуализации. В настоящее время почти каждая научная проблема связана с измерением параметров реальных объектов или с моделированием процессов, которые связаны с этими объектами. В обоих случаях есть аппаратное и программное обеспечение системы под названием *решатели,* которые управляют научным экспериментом и получают выходные данные в специальном формате. Эти данные должны быть проанализированы с помощью других систем или людьми. Как правило, выходные данные, предназначенные для анализа человеком, должны быть визуализированы с целью упростить анализ. Однако форматы выходных данных, а также интерфейсы решателей существенно отличаются из-за большого количества исследуемых объектов в разных областях науки. Часто это приводит к тому, что становится невозможным повторное использование визуализаторов с другими решателями, даже в той же области науки. Для решения этой проблемы было разработано приложение SciVi [1]. Эта система визуализации обладает средствами высокоуровневой настройки на сторонние источники данных при помощи средств онтологического инжиниринга [2].

Однако часто возникает необходимость дополнительной настройки на специфику решаемой задачи, когда исследователю требуется осуществить некоторую особую обработку данных, выдаваемых решателем. Чтобы освободить пользователя от необходимости вносить изменения в решатель или создавать

промежуточное программное обеспечение для такого рода фильтрации, было решено разработать и реализовать механизм настраиваемой предобработки данных на стороне SciVi, добавив в ее состав подсистему фильтров.

Исследования показывают, что для визуализации схемы в виде диаграммы потока данных [3] является оптимальным вариантом. Такое представление полностью отражает параллелизм выполнения операций и зависимость передаваемых данных. Преимущества такого вида диаграмм особенно важны для научной визуализации в силу того, что данных для наглядного представления человеку поступает много и ставится вопрос о наиболее эффективном способе фильтрации, в то время как предложенный метод оптимизирован для параллельных вычислений.

Набор фильтров и их свойства управляются знаниями в виде онтологии [2], и при пополнении или корректировке не требуется изменения исходного кода самой системы. Такие знания представляют собой онтологию, которая описывает допустимые типы данных и доступные фильтры, и хранится в виде ориентированного графа [4]. Кроме того, знания о фильтрах несут в себе не только описание, но и реализацию. В силу перечисленных выше причин, фильтры в системе обладают смысловым значением. Поэтому для обозначения предлагаемого подхода применяется обозначение *семантический фильтр.*

Для визуализации таких фильтров можно строить графический интерфейс пользователя на основе знаний в виде онтологий. В статье [5] описываются соответствующие механизмы по построению графического интерфейса пользователя, исходя из онтологических знаний. Выделяются несколько уровней опыта взаимодействия пользователя, представляющие собой восприятие и ответные действия пользователя, в соответствии с которыми самый абстрактный уровень модели – уровень стратегии – определяется онтологией и на его основе строится уровень поверхности, непосредственно с которым взаимодействует пользователь (одновременно он является интерфейсом пользователя).

Для того чтобы на уровне графического интерфейса пользователя позволить клиенту составить схему семантических фильтров с возможностью фильтрации на стороне сервера и избавить от необходимости написания промежуточного программного обеспечения для задачи фильтрации был принято решение о создании библиотеки. Для достижения этой цели необходимо решить следующие задачи:

1. построить формальную модель подсистемы семантических фильтров;
2. рассмотреть существующие программные решения для построения графического интерфейса;
3. разработать библиотеку, позволяющую пользователю настраивать систему предобработки данных на уровне графического интерфейса пользователя.

1. Формальная модель подсистемы семантических фильтров

Будем считать, что семантический фильтр представляет собой отображение φ вида

$$\varphi :< I, P >\rightarrow O,$$

где I – множество типизированных входов, P – множество настроечных параметров, O – множество типизированных выходов.

Например, фильтр *clamp*, ограничивающий входное значение между границами, имеет множеством I единственный элемент x численного типа. Множество P будет состоять из двух значений, отвечающих за минимальную и максимальную границы диапазона значений. Множество O состоит из одного элемента y, соответствующего значению x в заданном диапазоне.

Знания о предметной области хранятся в виде онтологий – формальных моделей, включающих в себя множество понятий этих предметных областей с их определениями, множество связей между понятиями и множество аксиом, введенных на этих понятиях и связях. Такой подход позволит в дальнейшем расширять функциональность без необходимости переписывания отлаженной программы. Поэтому для представления системы семантических фильтров необходима онтология F, описывающая допустимые типы данных и сведения о конвертации из одного типа данных в другой, входные и выходные параметры для каждого фильтра, допустимые для системы фильтры и ссылки на модули, реализующие эти фильтры.

Обозначим через K допустимое состояние системы. K описывается упорядоченной парой $K :< N, L >$, где N – множество добавленных на текущую схему семантических

фильтров, L – множество проведенных связей между фильтрами.

Каждый элемент множеств I и O соответствует одному из типов данных, представленных в онтологии F, которые могут быть приведены к другим типам данных в пределах заданных правил преобразования. Поэтому при добавлении связи в множество L должна осуществляться проверка на возможность преобразования типов данных, если они различны.

С учетом вышеизложенного, в данной работе предлагается хранить структуру схемы системы семантических фильтров в виде ориентированного графа G, который является стандартной формой представления онтологий [2].

$$G := (V, E),$$

где V – множество узлов графа, E – множество упорядоченных пар – ребер.

Множеству узлов V будет соответствовать множество фильтров N. А множество L будет представлено в виде ребер E. Таким образом, для генерации кода программы, построенной по данному графу G (позволяющей непосредственно осуществить фильтрацию данных), необходимо совершить его обход. С этой целью наиболее подходящим является поиск в глубину. Преимущество этого метода состоит в том, что исходная вершина обхода – обработанные данные системой фильтров – зависит от других вершин, которые в свою очередь зависят от других фильтров, и так далее. Таким образом, происходит удовлетворение всех зависимостей для того чтобы обработать входные данные.

2. Обзор решений по созданию графического интерфейса

Для решения задачи представления системы семантических фильтров в наглядном для пользователя виде необходимо разработать библиотеку с высокоуровневым графическим интерфейсом, которая должна удовлетворять следующим требованиям:

1. настройка фильтрации данных на стороне сервера SciVi должна осуществляться через web-интерфейс управления сервером;
2. редактор графов должен быть реализован как часть web-приложения, используя для своей работы CSS, HTML и JavaScript;
3. редактор графов должен работать как на десктопных, так и на мобильных устройствах, чтобы пользователь мог использовать систему с любого устройства;
4. необходима поддержка современных браузеров (Chrome, Firefox, Internet Explorer), чтобы избавить пользователя от необходимости устанавливать дополнительное программное обеспечение.

Чтобы удовлетворить данные требования, были проанализированы популярные библиотеки, предназначенные для построения графического web-интерфейса пользователя и отображения диаграмм, написанные на языке программирования JavaScript (см. табл.).

Сравнение средств построения графического web-интерфейса

Параметры сравнения	jQuery UI	Webix	D3.js	FabricJS
Высокоуровневый API	+	+	-	-
Лицензия	MIT	GNU GPL, Commercial	BSD	MIT
Настройка на предметную область	-	-	+	+
Создание диаграмм	-	-	+/-	+
Средства построения интерфейса	HTML, DOM	HTML, DOM	SVG	SVG
Поддерживаемые браузеры	все десктопные и мобильные	все десктопные и мобильные	все десктопные и мобильные	только десктопные

Таким образом, существующие решения можно разделить на два типа: фреймворки, предоставляющие множество средств для решения широкого круга задач, и низкоуровневые библиотеки, которые способны к высокопроизводительной визуализации. Но несмотря на перечисленные выше достоинства, существуют также проблемы, характерные для рассмотренных типов: фреймворки имеют многоуровневую архитектуру, из-за которой происходит падение производительности при манипулировании большим числом объектов; библиотеки визуализации предоставляют низкоуровневый API, вследствие чего необходимо дописывать функциональность, которая уже реализована производителями браузеров. В силу изложенных причин было принято решение разработать собственное решение.

3. Разработка графического интерфейса

С учетом вышесказанного, было принято решение о реализации подсистемы семантических фильтров в виде библиотеки, написанной на языке программирования JavaScript с использованием HTML и CSS, что позволит выполнять программу средствами программы-браузера и обеспечит кроссплатформенность решения.

Установлено, что реализация в качестве библиотеки позволит обеспечить минимально необходимую функциональность по построению и манипулированию графическими элементами пользователя. Это в свою очередь позволит избавиться от потери производительности в случае с частыми операциями манипуляции элементами, которые являются *узким местом* для фреймворков. Но такая библиотека будет предоставлять более высокоуровневый API для разработчика. Кроме того, разработанное приложение можно будет внедрить в любое другое, использующее web-браузер.

Построение графического интерфейса пользователя на основе HTML-элементов поможет избежать реализации уже существующего механизма манипуляции данными, который присутствует в браузере.

Кроме того, при разработке системы используются стандартные элементы HTML и код на языке JavaScript, который поддерживается наиболее популярными браузерами, что делает разрабатываемую библиотеку универсальной и позволяет включить его в клиент-серверное приложение SciVi с минимальной настройкой.

Как отмечалось выше, внутренняя схема подсистемы семантических фильтров представляет собой ориентированный граф.

Стоит отметить, что от такого графа не требуется наличие только одной компоненты связности. Так как такая схема графа будет передаваться на сервер в процессе построения для возможности сохранения, пользователь на момент передачи может не успеть объединить все узлы в одну компоненту связности.

Кроме того, пользователь может не применять получаемые программой данные для обработки, а использовать порождающие фильтры, такие как генератор случайных чисел, фильтры, дающие на выходе строку и другие.

Узел графа содержит информацию о всех параметрах фильтра, которому он соответствует. Любое изменение поведения фильтра на уровне графического интерфейса фиксируется на уровне бизнес-логики. Ребро устанавливает связь между параметрами и делает возможным обход графа для составления результирующей программы, соответствующей построенной схеме.

Поскольку в схеме всегда присутствуют узлы выходных и входных данных для схемы в общем, обход графа происходит с вершины выходных данных. По окончанию обхода схемы строится представление в формате JSON о том, какие фильтры, с какими параметрами, и каким образов необходимо применить для обработки данных.

Параметры в каждом узле хранятся в виде пары (параметр, значение). Язык программирования JavaScript является слабо типизированным, что позволяет хранить параметр как строковое значение и значение может принимать любой тип данных без лишнего приведения типов. Кроме значения хранится информация о том, является ли параметр входным (указывается значение *"Input"*) или выходным (указывается значение *"Output"*). При проходе графа собирается информация о всех параметрах для каждого фильтра, а при составлении программы на место соответствующих параметров подставляются их значения.

Хранение ребер в виде пары
выходной_параметр_фильтра_1,
входной_параметр_фильтра_2
позволяет определить, как именно связаны два фильтра (они могут быть связаны не одним ребром). При обходе эта информация учитывается и при составлении программы: вместо параметров одного фильтра подставляются результаты работы другого фильтра, с которым связан первый.

Результатом обхода графа является объектная нотация языка программирования JavaScript – JSON.

Этот формат хорош тем, что объект в нем представлен в виде пары *ключ:значение*, что соответствует описанию параметров фильтра и позволяет лаконичнее отобразить ребра. Кроме того, для чтения данных в таком формате существует множество библиотек для различных языков.

Структура графа в нотации JSON представлена в листинге.

```
{
 //Раздел описания фильтров
 "Filter1": {
    //Описание параметров фильтра и значений.
    //Ключ – название фильтра, значение – мас-
сив из непосредственно
    //значения параметра и информации о типе
параметра:
    //Input – параметр входной, Output – пара-
метр выходной.
    "Param1": ["Value1", "Input"],
    "Param2": ["Value2", "Output"],
    …
 },
 "Filter2": {
    "Param": ["Value", " Input "],
    …
 },

 //Раздел описания связей
 "Filter1_Filter2": {
    //Указываются какие параметры связаны
    "Filter1": "Param2",
    "Filter2": "Param"
 }
}
```

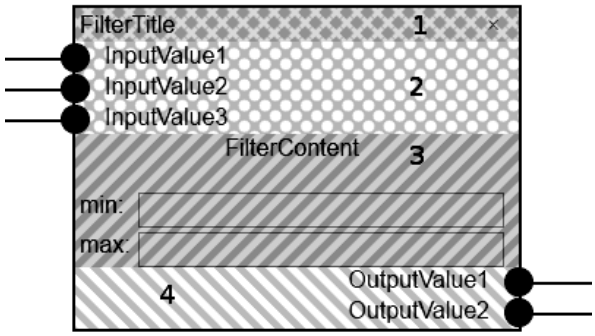
Листинг структуры графа в нотации JSON

Для возможности редактирования параметров фильтров, их представления в виде графических окон пользователя и реализации алгоритмов на динамическом языке программирования используется онтология. При попытке пользователя получить список всех доступных фильтров клиент обращается к базе знаний, хранящейся на сервере, и получает в ответ информацию о представлении параметров фильтров в виде HTML, CSS и JavaScript. При ошибке обращения к базе знаний пользователю выводится соответствующее сообщение об ошибке.

С целью обеспечения безопасности данных администратор должен предоставить доступ к базе в режиме чтения. При этом нет никакого гарантированного метода избежать выполнения зловредного кода, который может быть записан в базе знаний. Плюсом и одновременным минусом интерпретируемых языков является возможность выполнить любой код программы, записанной в строку [6]. За подобную функциональность зачастую отвечает функция eval [7], которая принимает выражение, записанное на языке, в котором она выполняется, в виде строки. Конечно, существуют более безопасные методы выполнения кода, но они значительно ограничивают возможности для программиста. Поэтому необходимо уделить внимание проблеме безопасности базы знаний. Это особенно важно, поскольку изменения в базе на сервере влияет на работу всех клиентов.

Одной из основных частей разрабатываемой подсистемы является графический интерфейс пользователя: графические элементы, ответственные за визуализацию фильтров и связей между ними.

Семантический фильтр представляется на схеме в виде графического окна, поддерживающего операции перемещения, изменения размера, закрытия. Общий вид окна показан на рисунке.



Пример тестового фильтра
Окно состоит из четырех основных областей: 1 – заголовок окна; 2 – входные параметры; 3 – область настроечных параметров; 4 – выходные параметры

Заголовок окна состоит из названия семантического фильтра, идентифицирующего его назначение, и кнопки закрытия окна. Область входных параметров содержит графические элементы, состоящие из названия входного параметра и специального элемента, отвечающего за протягивание связей между окнами. Область выходных параметров аналогична области входных параметров. Каждый элемент, отвечающий за визуализацию параметра фильтра, хранит информацию о том, к какому классу он принадлежит: входной или выходной. Это позволяет избежать ошибок при установлении связей между фильтрами. Основной контент размещается в области настроечных параметров. Так как эта область размечается на языке HTML, то такой подход придает гибкость в визуализации семантического фильтра. Этот контент может быть произвольным, что позволяет создавать окна для семантических фильтров различных типов.

Кроме того, возможно размещение элементов, предоставляемых сторонними библиотеками, например двухмерный график или средство выбора цвета.

Заключение

В данной статье описано направление развития адаптивной мультиплатформенной клиент-серверной системы научной визуализации SciVi, построенной на принципах онтологического инжиниринга, в виде подсистемы семантических фильтров. Семантические фильтры позволяют гибко настраивать SciVi на специфику решаемой задачи, освобождая пользователя от необходимости изменять исходный код решателя в том случае, если требуется оперативно по некоторому алгоритму преобразовать генерируемые данные перед визуализацией (например, применить к ним некоторую оконную функцию).

В статье была построена формальная модель для внутреннего представления информации о предложенной схеме фильтров. Был проведен сравнительный анализ существующих библиотек по созданию графического интерфейса пользователя и визуализации данных. Среди основных недостатков были выделены низкоуровневый API и низкая производительность при частом манипулировании объектами.

В результате была реализована библиотека на языке JavaScript с использованием CSS и HTML, что позволит использовать SciVi без установки пользователем дополнительного программного обеспечения (требуется лишь наличие браузера). Онтологический подход к настройке подсистемы заключается в том, что добавляется гибкость и возможность добавления семантических фильтров без необходимости изменения отлаженной программы.

Список литературы

1. *Ryabinin K., Chuprina S.* Development of Multiplatform Adaptive Rendering Tools to Visualize Scientific Experiments // Procedia Computer Science. Vol. 29. 2014. P. 1825–1834.
2. *Авдошин С.М., Шатилов М.П.* Онтологический инжиниринг // Бизнес-информатика. 2007. № 2. С. 32–36.
3. *Lee B., Hudson A.R.* Issues in Dataflow Computing // Advances in Computers. Vol. 37. 1993. P. 285–333.
4. *Mitra P., Wiederhold G., Kersten M.* A Graph-Oriented Model for Articulation of Ontology Interdependencies // Lecture Notes in Computer Science. Vol. 1777. 2000. P. 86–100.
5. Ontological Model Driven GUI Development: User Interface Ontology Approach URL: https://goo.gl/V4YDzw (дата обращения: 02.07.2016).
6. *Summerfield M.* Programming in Python 3: A Complete Introduction to the Python Language (2nd Edition). Addison-Wesley Professional. 2009.
7. eval() – JavaScript |MDN URL: https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/eval (дата обращения: 01.06.2016).

Semantic filters for input data for the scientific visualization system SciVi

A. Yu. Bortnikov

Perm State University; 15, Bukireva st., Perm, 614990, Russia

a.yu.bortnikov@yandex.ru; +7-912-49-241-09

This article deals with the method of extension of the scientific visualization system SciVi, which can be fine-tuned to the specifics of problems solved in this system. The method of constructing semantic filters by means of a user's graphical interface is suggested. Extensibility and flexibility of the solution proposed are achieved through controlling the program operation by means of ontological knowledge base on filters. To adjust the visualization system to the specifics of a particular task, the user is only to connect nodes which describe input and output data of a solver with objects selected for the visualization. The semantic filters that are available to the user from the repository serve as intermediate nodes in the resulting data flow diagram. The client-server architecture of the application allows for distributing the load in accordance with the users' needs.

Keywords: *scientific visualization; data flow; semantic filter; ontological engineering; data filtering.*