



Министерство науки и высшего образования Российской Федерации
федеральное государственное бюджетное образовательное
учреждение высшего профессионального образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Робототехники и комплексной автоматизации»

КАФЕДРА «Системы автоматизированного проектирования (РК-6)»

РАСЧЁТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

по дисциплине «Технологии интернет»

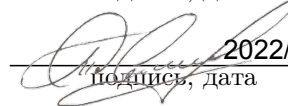
на тему

«Разработка программных средств описания состояний данных в рамках
реализации графоориентированного подхода»

Студент РК6-81Б
группа

Руководитель КП

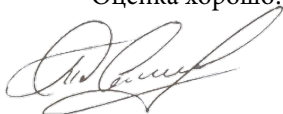
подпись, дата

 2022/06/09
подпись, дата

Тришин И.В.
ФИО

Соколов А.П.
ФИО

Оценка хорошо!



А.П. Соколов

Москва, 2022

« » 2022 г.

на выполнение курсового проекта

(фамилия, имя, отчество)

 подпись, дата
 подпись, дата

Соколов А.П.
ФИО

РЕФЕРАТ

курсовой проект: 14 с., 7 глав, 5 рис., 0 табл., 6 источн.

ВИЗУАЛИЗАЦИЯ АССОЦИАТИВНЫХ МАССИВОВ, СТРУКТУРЫ ДАННЫХ, МЕТАДАННЫЕ, ПРЕДМЕТНО-ОРИЕНТИРОВАННЫЕ ЯЗЫКИ ПОГРАММИРОВАНИЯ.

В современных системах, осуществляющих многоэтапные вычисления, существует потребность в средстве визуализации обрабатываемых данных. Данная работа направлена на разработку информационных программных структур, описывающих обрабатываемые данные с их типами и возможными пояснениями, и подбор некоторого формата, в котором описания данных могли бы храниться на постоянном запоминающем устройстве.

Тип работы: курсовой проект.

Тема работы: *«Разработка программных средств описания состояний данных в рамках реализации графоориентированного подхода».*

Объект исследования: методы и технологии программного описания многомерных ассоциативных массивов.

Основная задача, на решение которой направлена работа: разработка информационных компонентов средств визуализации многомерных ассоциативных массивов.

Цель работы: предложить программные архитектурные решения для создания программных средств визуализации состояний данных в контексте применения графоориентированного подхода

В результате выполнения работы: 1) изучены форматы представления разнотипной научной информации; 2) изучены теоретические основы “графоориентированного подхода”, введено понятие состояния данных; 3) разработаны структуры данных, поддерживающее хранение информации о состояниях данных;

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1. Постановка задачи	7
1.1. Концептуальная постановка задачи	7
2. Программная реализация	9
2.1. Архитектура	9
ЗАКЛЮЧЕНИЕ	13
Литература	14

ВВЕДЕНИЕ

При разработке программных систем, оперирующих большими объёмами данных, как, например, системы автоматизированного проектирования (CAD) и инженерного анализа (CAE) необходимо предусмотреть возможность предоставления результатов моделирования пользователю в доступном для него виде. Главным инструментом, реализующим такую возможность, можно считать средство визуализации. Основной её идеей является предоставление большого объёма информации исследователю в форме, удобной для восприятия и анализа [1]. Примером такого анализа может быть поиск закономерностей влияния одних переменных на другие. Таким образом, визуализация поддерживает принятие решений [2].

Кроме того, существует потребность помимо самих данных отображать некоторую связанную с ними мета-информацию, которая служит для их документирования. Действительно, для удобства восприятия и последующего анализа исследователю необходимо понимать, что представляет тот или иной элемент визуализируемых данных. Таким образом, необходимо некоторое средство создания и связывания мета-информации с данными.

В ходе программной реализации сложной системы (например, системы инженерного анализа), инструмент визуализации является отдельной подсистемой, имеющей некоторый программный интерфейс взаимодействия со всей остальной системой. Логика работы системы с средством визуализации представлена на рисунке В.1.

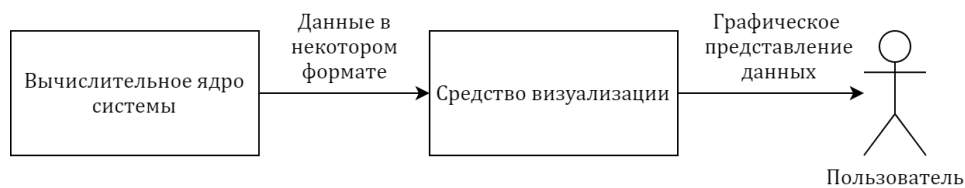


Рисунок В.1. Схема работы средства визуализации

Примерами реализации таких инструментов может служить отдельное веб-приложение, получающее данные от сервера, где проводятся основные вычисления [3], или отдельный компонент, отвечающий за формирование и вывод графического представления данных, в подсистеме взаимодействия с пользователем. На рисунке В.1 показано, что данные в средство визуализации передаются в некотором формате. Это может быть как их двоичное внутреннее представление в оперативной памяти, все компоненты системы имеют общее адресное пространство, или некоторый файл, где данные структурированы особым образом.

В распределённой вычислительной системе GCD средство визуализации данных является отдельным программным модулем. От вычислительного ядра системы данные должны передаваться через файлы. Таким образом, одной из задач исследования было подобрать (или при необходимости разработать) наиболее подходящий формат представления данных для их передачи в средство визуализации.

Одними из первых рассмотренных были форматы для хранения научных данных HDF4 и HDF5 [4]. Данные бинарные форматы позволяют хранить большие объёмы гетерогенной ин-

формации и поддерживают иерархическое представление данных. В нём используется понятие набора данных (англ. *dataset*), которые объединяются в группы (англ. *group*). Кроме того, формат HDF5 считается «самодокументирующимся», поскольку каждый его элемент – набор данных или их группа – имеет возможность хранить метаданные, служащие для описания содержимого элемента. Существует официальный API данного формата для языка C++ с открытым исходным кодом. Одним из главных недостатков HDF5 является необходимость дополнительного ПО для просмотра и редактирования данных в этом формате, поскольку он является бинарным.

Альтернативой бинарным форматам описания данных являются текстовые. Среди них были рассмотрены форматы XML (Extensible Markup Language) и JSON (Javascript Object Notation). Главным преимуществом формата XML является его ориентированность на древовидные структуры данных и лёгкость лексико-синтаксического разбора файлов этого формата. Среди недостатков стоит выделить потребность в сравнительно большом количестве вспомогательных синтаксических конструкций, необходимых для структурирования (тегов, атрибутов). Они затрудняют восприятие чистых данных и увеличивают итоговый объём файла.

Формат JSON, так же, как и XML рассчитан на иерархические структуры данных, но является не столь синтаксически нагруженным, что облегчает восприятие информации человеком [5]. Кроме того, крайне важным преимуществом JSON является его поддержка по-умолчанию средствами языка программирования Javascript, который используется при разработке веб-приложений. При этом JSON также обладает рядом недостатков. Среди них сниженная, по сравнению с XML надёжность, отсутствие встроенных средств валидации и отсутствие поддержки пространств имён, что снижает его расширяемость.

На основании проведённого анализа преимуществ и недостатков выбор был сделан в пользу формата JSON. Ключевыми факторами для этого стали лёгкость восприятия информации в этом формате и нативная поддержка этого формата языком Javascript.

1 Постановка задачи

1.1 Концептуальная постановка задачи

В ходе выполнения работы необходимо было реализовать программные инструменты описания состояний данных согласно их определению, данному в [6]. Должна также быть реализована возможность сохранять описания состояний данных в файлы с использованием некоторого формата.

Определение 1. Множеством элементарных состояний данных сложного вычислительного метода (СВМ) будем называть такое множество \mathbb{W} пар “имя параметра – множество допустимых значений параметра” таких, что $\mathbb{W} = Ind(string) \times \bigcup_{i=0}^{N_T} \{Ind(T_i)\}$, где $Ind(X)$ – оператор индукции, определяющий множество всех возможных значений типа X , $string$ – тип строк в заданном языке программирования и T_i – i -тый произвольный тип заданного языка программирования, N_T – число поддерживаемых типов данных в заданном языке программирования (при поддержке языком пользовательских типов N_T бесконечно).

Элементами множества элементарных состояний СВМ могут служить следующие пары:

$$s_a = (\langle RealParam \rangle, \mathbb{R}),$$

$$s_b = (\langle IntegerParam \rangle, \mathbb{Z}),$$

$$s_c = (\langle MethodName \rangle, Ind(string)),$$

где $s_a, s_b, s_c \in \mathbb{W}$.

Определение 2. Пространством состояний СВМ \mathbb{S} будем называть множество таких подмножеств $S_i \subset \mathbb{W}$, что элементы каждого такого подмножества имеют уникальные имена в рамках этого подмножества, т.е. $\forall s_1, s_2 \in S_i : s_1 \neq s_2 \Rightarrow pr_1(s_1) \neq pr_1(s_2)$, где $pr_1(X)$ – проекция декартова произведения на первую координату соответствующей пары.

Определение 3. Состоянием СВМ будем называть элемент $S \in \mathbb{S}$

Определение 4. Пусть $S \in \mathbb{S}$ – состояние СВМ. Тогда множество D , такое что $\forall s \in S \exists d \in D : d = (n, v), n = pr_1(s), v \in pr_2(s)$, будем называть данными сложного вычислительного метода в состоянии S .

Ниже приведены примеры данных в некоторых состояниях:

$$D_1 \multimap S_1 = \{(\langle RealParam \rangle, 1.0), (\langle StringParam \rangle, \langle Hello, world! \rangle)\},$$

$$S_1 = \{(\langle RealParam \rangle, \mathbb{R}), (\langle StringParam \rangle, Ind(string))\},$$

$$D_2 \multimap S_2 = \{(\langle N \rangle, 5), (\langle M \rangle, 3)\},$$

$$S_2 = \{(\langle N \rangle, \mathbb{Z}), (\langle M \rangle, \mathbb{Z})\}$$

Данные в соответствующем состоянии, как правило, удобно хранить в виде ассоциативного массива. При реализации структуры данных для внутреннего представления состояний данных необходимо для каждого элемента состояния хранить его имя и тип. Помимо этого в

целях повышения удобства восприятия данных должна быть включена возможность добавить к каждому элементу состояния данных краткое описание для пояснения роли конкретного входного параметра или промежуточной переменной в реализации вычислительного метода.

Тип отдельной переменной может быть как скалярным (целое, логическое, вещественное с плавающей запятой и пр.), так и сложным «векторным» (структурой, классом, массивом и пр.). Примером сложного «векторного» типа является, в свою очередь, ассоциативный массив со строковыми ключами, при этом конкретная переменная этого типа будет хранить, как правило, адрес этого массива. В общем случае элементы данного массива могут иметь разные типы. В рассматриваемом случае возникает возможность организации хранения состояния данных в виде иерархических структур.

Таким образом, для описания состояний данных требуется формат, который бы поддерживал гетерогенные (т.е. разнотипные) иерархические структуры данных.

2 Программная реализация

2.1 Архитектура

2.1.1 Описание элемента состояния данных

Текущая реализация “графоориентированного подхода” в библиотеке comsdk поддерживает обработку данных следующих типов:

- целые числа,
- числа с плавающей точкой,
- строки,
- числовые векторы,
- ассоциативные массивы с строковыми ключами.

Для их представления в разрабатываемой библиотеке был объявлен перечислимый тип `DType` (рис. 2).

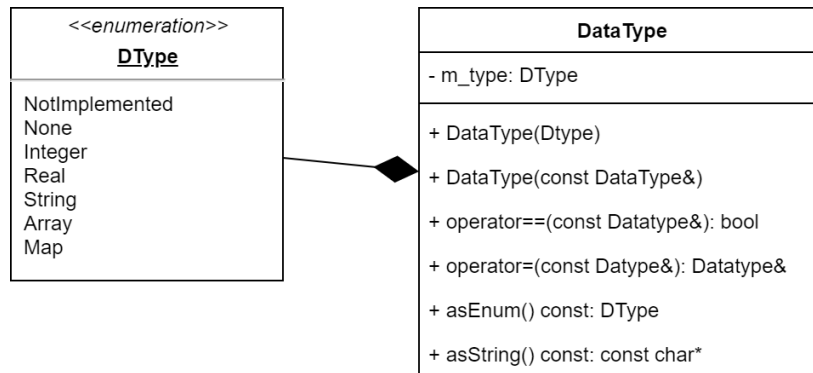


Рисунок 2. UML-диаграмма структуры данных для описания типов используемых данных

Помимо перечисленных выше типов присутствуют два дополнительных: `None` и `NotImplemented`. Данные типы используются для обозначения ошибок и исключительных ситуаций.

Класс `DataType`, представленный на рисунке 2 хранит только числовую константу, обозначающую тип и реализует некоторую дополнительную функциональность, а именно:

- даёт возможность присваивать и сравнивать типы при помощи соответствующих перегруженных операторов;
- даёт возможность получить имя типа при помощи метода `asString()`.

Для каждого из поддерживаемых на данный момент типов данных был создан константный объект класса `DataType`, содержащий в себе описание этого типа. Константы имеют следующие имена: `dInt`, `dReal`, `dString`, `dVector`, `dMap`. Кроме того, были объявлены две дополнительные константы для описания ситуации, когда запрашиваемый элемент не найден – `dNone` – и для ситуации, когда у запрошенной операции отсутствует реализация – `dNotImplemented`.

На рисунке 3 представлена UML-диаграмма основного класса, описывающего элемент состояния данных.

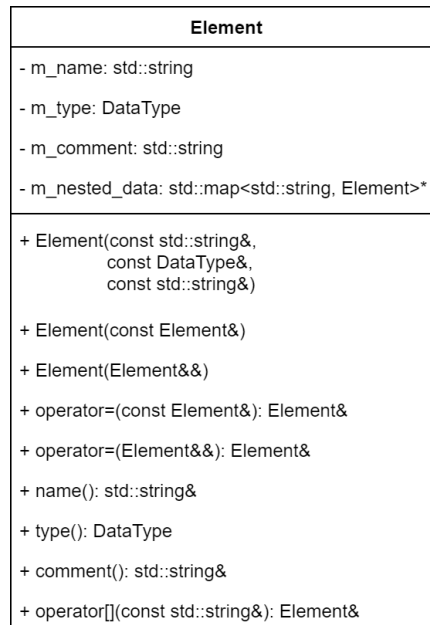


Рисунок 3. UML-диаграмма структуры данных для описания элемента состояния данных

В классе `Element` поля `m_name`, `m_type` и `m_comment` хранят в себе имя элемента, его тип (целочисленный, с плавающей точкой, строковый, числовой векторный, ассоциативный массив) и краткое описание соответственно. Закрытое поле `m_nested_data` содержит указатель на ассоциативный массив объектов типа `Element`. Данное поле используется, если сам элемент имеет тип ассоциативного массива (`dMap`). Поскольку элементы такого массива могут иметь разные типы, целесообразно для каждого такого элемента хранить их ключи и типы. Таким образом, каждый элемент ассоциативного массива по своей структуре повторяет элемент состояния данных. Это свойство позволяет делать состояния данных иерархическими. Использование указателя на объект `std::map` позволяет эффективнее использовать память в случае, когда тип элемента отличен от ассоциативного массива и нет необходимости обеспечивать доступ к его внутренним элементам.

Конструктор класса принимает на вход две строки и объект класса `DataType`, которые соотносятся с полями `m_name`, `m_comment` и `m_type` соответственно. Были явно реализованы конструктор копирования и оператор присваивания, поскольку в классе используются указатели на память, за выделение и освобождение которой отвечает он сам.

Для работы с элементами ассоциативного массива был реализован метод доступа по ключу (`operator[]`). При отсутствии элемента с запрошенным ключом в массиве он должен создаваться. Во избежание неопределённого поведения (англ. *undefined behaviour*) при вызове данного метода у элемента, тип которого отличен от ассоциативного массива (`Map`), метод будет возвращать специальную константу `NotImplemented` с одноимённым типом и пустым именем. Кроме того, для `const`-версии этого метода была добавлена проверка на наличие элемента в

массиве. В случае его отсутствия будет возвращена константа `None`, имеющая одноимённый тип и пустую строку в качестве имени.

2.1.2 Описание состояния данных

На рисунке 4 представлена UML-диаграмма разработанного класса, отвечающего за представление всего состояния данных вычислительного метода.

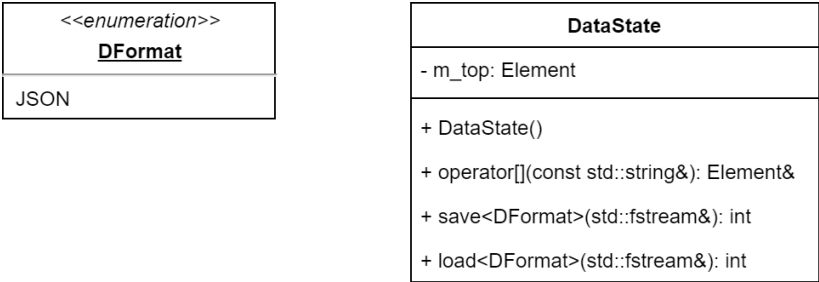


Рисунок 4. UML-диаграмма класса, представляющего состояние данных

Единственным закрытым членом этого класса является объект класса `Element`, который является “корневым” элементом состояния данных. Он имеет имя “root” и всегда имеет тип `Map`, и все элементы самого состояния данных хранятся внутри корневого элемента. При этом их ключи совпадают с их именами. Это даёт интуитивный интерфейс для взаимодействия с иерархической структурой состояния данных.

В классе `DataState` объявлены два шаблона метода загрузки состояния данных из файла и его сохранения в файл. Параметром шаблона является переменная перечислимого типа `DFormat`, в котором определены возможные файловые форматы, с которыми возможно взаимодействие. Подобный подход позволяет создавать различные реализации загрузки и сохранения состояний данных с сохранением единого интерфейса, что положительным образом сказывается на расширяемости разработанной кодовой базы.

Общая структура разработанных классов представлена на рисунке 5

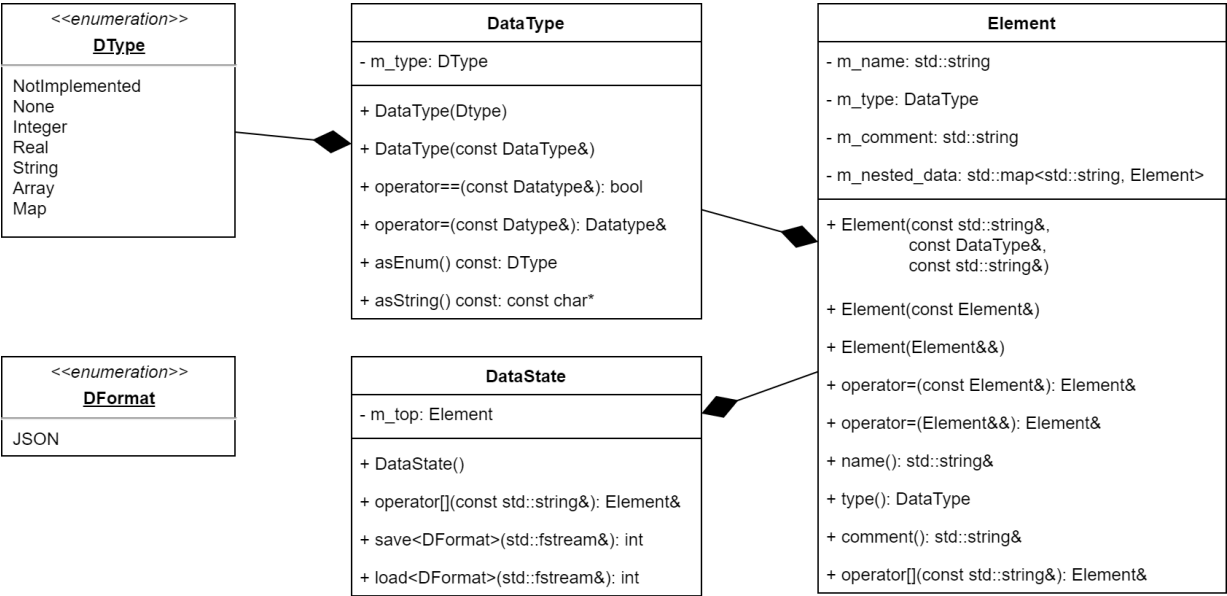


Рисунок 5. Структура разработанных классов

ЗАКЛЮЧЕНИЕ

Таким образом, в результате выполнения данной работы были разработаны программные инструменты для описания состояний данных сложных вычислительных методов. Данные средства дают как представление состояний данных в оперативной памяти, так и возможность сохранения и загрузки их описаний из файлов формата JSON.



В дальнейшем разработанные инструменты войдут в состав средства визуализации состояний данных, которое будет применяться при работе с вычислительной системой GCD.

Список использованных источников

- 1 Берченко Д.А., Круг П.Г. Аналитический обзор методов визуализации данных // Евразийский научный журнал. 2017. № 5.
- 2 И.К. Романова. Современные методы визуализации многомерных данных: анализ, классификация, реализация, приложения в технических системах // Наука и Образование. МГТУ им. Баумана. 2016. № 3. С. 133–167.
- 3 An overview of visualization and visual analytics applications in water resources management / Haowen Xu, Andy Berres, Yan Liu [и др.] // Environmental Modelling and Software. 2022. Т. 153.
- 4 The HDF Group [Электронный ресурс] [Оф. сайт]. 2022. Дата обращения: 17.05.2022. URL: <https://www.hdfgroup.org>.
- 5 JSON vs XML: What's the difference? | Guru99 [Электронный ресурс]. 2022. Дата обращения: 18.05.2022. URL: <https://www.guru99.com/json-vs-xml-difference.html>.
- 6 Соколов А.П. Першин А.Ю. Графоориентированный программный каркас для реализации сложных вычислительных методов // Программирование. 2018. № X.

Выходные данные

Тришин И.В.. Разработка программных средств описания состояний данных в рамках реализации графоориентированного подхода по дисциплине «Технологии интернет». [Электронный ресурс] — Москва: 2022. — 14 с. URL: <https://sa2systems.ru:88> (система контроля версий кафедры РК6)

Постановка:  канд. физ.-мат. наук, Соколов А.П.
Решение и вёрстка:  студент группы РК6-81Б, Тришин И.В.

2022, весенний семестр