Компьютерные инструменты в образовании, 2025
№ 1: 91–107
УДК: 004.822
http://cte.eltech.ru
doi:10.32603/2071-2340-2025-1-91-107

информатика

ЯЗЫК ОПИСАНИЯ ОНТОЛОГИЙ ONTOL V1

Аптуков М. И.[1], студент, aptukovm@yandex.ru
Марков М. Д.[1], студент, mike.markov04@gmail.com
Новиков Ф. А.[1], доктор техн. наук, профессор, fedornovikov51@gmail.com, orcid.org/0000-0003-4450-0173
Пестряков Д. Д.[1], аспирант, ассистент, ✉ pestryakov.dd@edu.spbstu.ru, orcid.org/0009-0000-6877-5716
Скворцов В. С.[1], студент, mail@vladimirskvortsov.com, orcid.org/0009-0001-0056-4805
Хамидуллин И. И.[1], студент, ilsaf.corp@gmail.com

[1]Санкт-Петербургский политехнический университет Петра Великого, ул. Политехническая, д. 8 лит. Б, 195251, Санкт-Петербург, Россия

Аннотация

В данной работе рассматривается язык описания онтологий ONTOL V1: его концепция, структура и возможности. Разработанный язык предназначен для формализации знаний и их структурированного представления, которое опирается на три ключевых компонента: определение типов (сущностей предметной области), описание функциональных зависимостей и задание иерархических отношений. Подчеркивается масштабируемость представления знаний на языке ONTOL V1, которая достигается за счёт модульной организации описаний и и возможности их импорта в другие модули, описанные на языке ONTOL V1. Ключевые цели использования языка включают наглядную визуализацию онтологических моделей, их сериализацию для интеграции в другие приложения, а также обеспечение основы для последующей машинной обработки. Рассматриваются потенциальные области применения ONTOL V1, прежде всего в образовательном процессе, а также в более широком контексте задач представления и автоматизированного использования знаний.

Ключевые слова: *онтология, предметная область, грамматики, образование, формализация знаний.*

Цитирование: Аптуков М. И., Марков М. Д., Новиков Ф. А., Пестряков Д. Д., Скворцов В. С., Хамидуллин И. И. Язык описания онтологий ONTOL V1 // Компьютерные инструменты в образовании. 2025. № 1. С. 91–107. doi:10.32603/2071-2340-2025-1-91-107

1. ВВЕДЕНИЕ

Предшествующие работы в рамках курса «Дискретная математика для программистов» в СПбПУ показали перспективность использования онтологий, представленных графически в нотации UML [1]. Однако у данного подхода есть и недостатки: слабая машиночитаемость и как следствие трудности при использовании онтологий для проверки утверждений с учетом ограничений предметных областей; изготовление больших онтологий с помощью визуальных средств редактирования может потребовать

больше времени, чем если бы использовалось текстовое представление; сложности при переиспользовании компонент. Для преодоления данных сложностей в рамках курса «Грамматики и автоматы» зимой-весной 2025 года был разработан язык представления онтологий ONTOL V1, диаграмма использования которого представлена на рисунке 1.

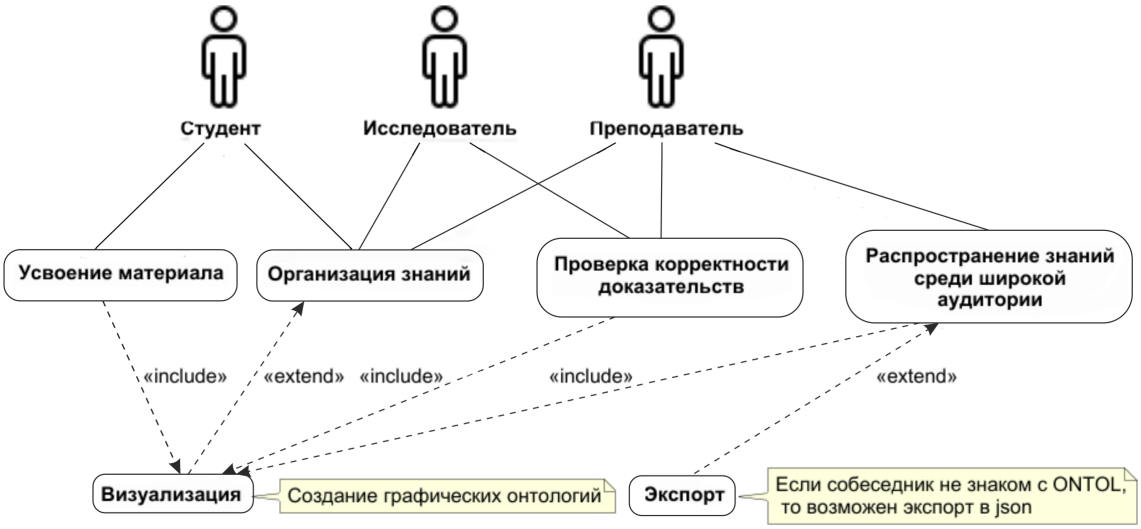

Рис. 1. Диаграмма использования языка Ontol V1

Действующие лица на диаграмме (рис. 1):

- Преподаватель — проектирует образовательный материал, формализует знания в онтологиях и корректирует траектории обучения.
- Исследователь — разрабатывает методы автоматизации, расширяет онтологический каркас предметных областей и анализирует эффективность системы.
- Студент — использует онтологии для решения лабораторных работ, изучает связи между темами и понятиями курса с помощью интерактивных графических схем.

Разработанный инструментарий предоставляет пользователю следующие варианты использования:

- Сериализация онтологий, записанных на ONTOL V1, в формат JSON (JavaScript Object Notation).
- Трансляция текста с языка ONTOL V1 на язык plantUML [2].
- Разбиение представления онтологий на несколько файлов (модульность).
- Выбор элементов онтологии для визуализации и непосредственно визуализация с помощью plantUML.

Данная статья сосредоточена на описании языка ONTOL V1 и разработанного инструментария для работы с ним, который позволяет создавать, визуализировать и использовать формализованные описания онтологий в образовательном процессе и за его пределами.

2. ИСТОРИЧЕСКИЙ ОБЗОР

Понятие онтологии как способа систематизации знаний имеет глубокие корни, однако его применение в информатике началось с развитием вычислительной техники

и возникновением потребности в формальных методах представления знаний для машинной обработки. В середине XX века появление вычислительных машин и последующее развитие систем управления базами данных, в частности реляционной модели данных в 1970-х годах [3], позволили эффективно хранить и обрабатывать большие объемы структурированной информации. Тем не менее, реляционные и многие другие ранние модели хранения данных имели существенный недостаток: они слабо отражали или вовсе не учитывали семантику данных — их значение и взаимосвязи в контексте предметной области.

Проблема явного представления семантики стала особенно острой с развитием и повсеместным распространением Всемирной паутины. Огромные объемы неструктурированной и полуструктурированной информации потребовали новых подходов для ее осмысления и автоматизированной обработки, в частности, для повышения релевантности поиска и интеграции данных из различных источников. Ответом на этот вызов стала концепция Семантической паутины (Semantic Web) [4], предполагающая создание единой системы стандартов для представления данных и знаний в машиночитаемом формате.

Под эгидой Консорциума Всемирной Паутины (W3C) были разработаны ключевые языки для представления онтологий в веб-среде. Первым шагом стала модель RDF (Resource Description Framework) [5], основанная на представлении знаний в виде триплетов «субъект-предикат-объект», формирующих ориентированный граф данных. Для описания словарей и простых таксономий поверх RDF была разработана RDF Schema (RDFS) [6], позволяющая определять классы сущностей и свойства (отношения). Дальнейшим развитием стал язык веб-онтологий OWL (Web Ontology Language) [7], построенный на основе RDFS и дескрипционной логики. OWL предоставил значительно большую выразительную мощность для описания сложных аксиом, ограничений и логических взаимосвязей между сущностями, став де-факто стандартом для построения формальных онтологий во многих областях знаний [8].

Параллельно с развитием текстовых языков Семантической паутины, для целей моделирования, в том числе и концептуального, активно использовались графические нотации. В частности, унифицированный язык моделирования UML (Unified Modeling Language) [9], изначально созданный для проектирования программных систем, нашел применение и для визуального представления онтологий, в том числе в образовании [1]. Диаграммы классов UML позволяют наглядно отображать сущности, их атрибуты и различные типы отношений (ассоциации, агрегации, наследование), что может быть полезно для понимания структуры предметной области человеком.

Однако и текстовые языки типа OWL, и графические подходы, такие как UML, имеют свои ограничения в контексте задач, решаемых в данной работе. Язык OWL, обладая высокой выразительностью, имеет высокий порог входа и может быть избыточно сложным для создания и интуитивного восприятия онтологий в образовательных целях, а также требует специальных инструментов для редактирования и визуализации. Графический язык UML, напротив, будучи интуитивно понятным визуально, по своей природе является стандартом обозначений и плохо приспособлен для автоматизированной обработки, анализа и использования в системах, требующих формального машиночитаемого представления знаний. Именно этот разрыв между потребностью в формализации для автоматизации (что сложно достичь с помощью UML) и необходимостью в простоте и наглядности для образовательного процесса (что может быть затруднительно с помощью OWL) послужил основной мотивацией для разработки языка ONTOL V1.

3. СИНТАКСИС ЯЗЫКА

Язык ONTOL V1 предназначен для формализации знаний и представления онтологий. Будучи декларативным, он позволяет разработчикам сосредоточиться на содержании, используя интуитивно понятный синтаксис с ключевыми блоками для структурного представления данных. Такая блочная структура упрощает работу с онтологиями, разделяя аспекты описания и отображая разнообразные данные и их взаимосвязи, что делает ONTOL V1 мощным инструментом представления знаний.

3.1. Основные элементы

Грамматика языка ONTOL V1 использует стандартные лексические элементы. Основные токены можно описать в РБНФ (Расширенная форма Бэкуса-Наура) следующим образом:

```
Цифра ::= "0" | "1" | ... | "9"
Буква ::= "a" | "b" | ... | "z" | "A" | "B" | ... | "Z"

Идентификатор ::= (Буква | "_") { Буква | Цифра | "_" }

СтроковыйЛитерал ::= "'" { ЛюбойСимволКромеОдинарнойКавычки } "'"
                    | '"' { ЛюбойСимволКромеОдинарнойКавычки } '"'

ПараАтрибутов ::= Идентификатор ":" СтроковыйЛитерал
Атрибуты ::= "{" { ПараАтрибутов } "}"

EOL ::= ПереводСтроки | КонецСтроки
```

3.2. Блок типов types

Блок `types:` используется для определения типов сущностей онтологии. В одном файле может быть несколько таких блоков, блок также может быть пустым. Формально структура блока и определений типов описывается так:

```
БлокТипов ::= "types:" { ОпределениеТипа }
ОпределениеТипа ::= Идентификатор ":"
СтроковыйЛитерал "," СтроковыйЛитерал [ "," Атрибуты ] EOL
```

Пример задания типа с меткой и описанием:

```
types:
student: 'Студент', 'Учащийся высшего учебного заведения'
teacher: 'Преподаватель', 'Лицо, преподающее учебный предмет'
```

Пример задания типа с атрибутами:

```
types:
exam: 'Экзамен', 'Форма проверки знаний', { color: '#FFCCCC' }
```

Возможные атрибуты:
- `color` — цвет прямоугольника.
- `note` — заметка, текст которой отобразится рядом с прямоугольником.

3.3. Блок функций functions

Блок functions: предназначен для описания функциональных зависимостей или процессов. В файле может быть несколько блоков ‘functions:‘, они могут быть пустыми. Синтаксис определения функции:

```
БлокФункций ::= "functions:" { ОпределениеФункции }
ОпределениеФункции ::= Идентификатор ":" СтроковыйЛитерал ","
                       "(" [ СписокАргументовВхода ] ")" "->" АргументВыхода
                       [ "," Атрибуты ] EOL
СписокАргументовВхода ::= Аргумент { "," Аргумент }
АргументВыхода ::= Аргумент
Аргумент ::= Идентификатор ":" СтроковыйЛитерал
```

Определение функции начинается с ее идентификатора, затем через двоеточие указывается метка (название в онтологии), далее через запятую описываются входные аргументы в круглых скобках (формат: ‘имя: ’тип/описание’‘), затем стрелка ‘->‘ и описание выходного аргумента (в том же формате). Опционально можно указать атрибуты для визуализации.

Пример задания функции:

```
functions:
pow: ’power’, (value: ’base’, value: ’degree’) -> value: ’result’
```

Пример задания функции с атрибутами:

```
functions:
gradeExam: ’Оценить экзамен’, (
    ex: ’exam’,
    score: ’integer’,
) -> result: ’grade’, {
    color: ’#green’,
    colorArrow: ’#blue’,
    inputTitle: ’Входные данные’,
    outputTitle: ’Результат’,
}
```

Возможные атрибуты функции:

- color — цвет прямоугольника,
- colorArrow — цвет стрелок,
- type — тип стрелок,
- inputTitle — описание входных стрелок,
- outputTitle — описание выходной стрелки.

3.4. Блок иерархий hierarchy

Блок «hierarchy:» используется для определения структурных отношений между типами (сущностями). В файле может быть несколько таких блоков, они могут быть пустыми. Синтаксис определения иерархической связи:

```
БлокИерархий ::= "hierarchy:" { ОпределениеСвязи }
ОпределениеСвязи ::= [ Идентификатор ":" ] Триплет [ "," Атрибуты ] EOL
Триплет ::= Идентификатор КлючевоеСловоОтношения Идентификатор
КлючевоеСловоОтношения ::= "dependence" | "association" | "directAssociation" |
                           "inheritance" | "implementation" | "aggregation" |
                           "composition"
```

Задание триплетов (связей) происходит двумя способами:

1. Анонимная связь: последовательно указываются идентификатор родительского типа, ключевое слово, обозначающее тип отношения, и идентификатор дочернего типа.
2. Именованная связь: сначала указывается идентификатор самой связи, затем двоеточие, а далее структура повторяет анонимную связь.

Опционально можно задать атрибуты для настройки визуализации связи.
Типы отношений (ключевые слова):

- `dependence` — временная связь, использование в методе.
- `association` — постоянная связь между классами.
- `directAssociation` — прямая связь без уточнений.
- `inheritance` — наследование свойств и поведения.
- `implementation` — реализация интерфейса (контракта).
- `aggregation` — слабое владение (часть-целое, может существовать отдельно).
- `composition` — сильное владение (часть не существует без целого).

Примеры задания связей:

```
hierarchy:
teacher inheritance person
university aggregation faculty
enrollment: student association course
```

Пример связи с атрибутами:

```
hierarchy:
department composition chair, { color: '#red', title: 'включает' }
courseToExam: course directAssociation exam, {
    direction: 'forward',
    leftChar: '1',
    rightChar: '0..*',
}
```

Возможные атрибуты связи:

- `color` — цвет стрелки,
- `direction` — направление стрелки (‘up‘, ‘down‘, ‘left‘, ‘right‘),
- `title` — подпись над стрелкой,
- `leftChar` — символ у начала стрелки (со стороны родительского типа),
- `rightChar` — символ у конца стрелки (со стороны дочернего типа).

3.5. Мета-информация

Мета-информация о файле онтологии задается в виде пар ключ-значение в начале файла (или в специальном блоке `meta:`, если он будет введен). Текущая версия языка предполагает следующие ключи:

```
БлокМета ::= { ПараМета }
ПараМета ::= КлючМета ":" СтроковыйЛитерал EOL
КлючМета ::= "version" | "title" | "author" | "description"
```

- `version` — номер версии файла онтологий,
- `title` — наименование файла онтологий,
- `author` — автор файла онтологий,
- `description` — описание файла онтологий.

Пример блока мета-информации:

```
version: '1.0'
title: 'Ontology Example'
author: 'Ivan Ivanov'
description: 'An example ontology description'
```

3.6. Блок объединения figure

Блок `figure` позволяет определить именованное подмножество (срез) онтологии для целей визуализации или экспорта только части общей модели. В одном файле может быть несколько блоков `figure`. Синтаксис:

```
БлокFigure ::= "figure" СтроковыйЛитерал ":" { Идентификатор } EOL
```

Сначала указывается ключевое слово `figure`, затем имя среза в виде строкового литерала, двоеточие, а затем с новых строк перечисляются идентификаторы элементов (типов, функций, именованных связей), которые должны войти в этот срез.

```
figure 'Основная структура':
employee
manager
project
task
relName # Именованная связь между project и client
assignTask # Функция
```

3.7. Импортирование import

Конструкция `import` позволяет использовать определения из других файлов онтологий, поддерживая модульность. Формальный синтаксис:

```
ОператорИмпорта ::= "import" ( "*" | "{" СписокИмпорта "}" )
                    "from" СтроковыйЛитерал EOL
СписокИмпорта ::= ЭлементИмпорта { "," ЭлементИмпорта }
ЭлементИмпорта ::= Идентификатор [ "as" Идентификатор ]
```

Язык поддерживает:

1. Импорт всех определений из внешнего файла (‘import * from ...‘).
2. Импорт конкретных определений (‘import name1, name2 from ...‘).
3. Переименование импортируемых определений с помощью ключевого слова ‘as‘.
4. Использование абсолютных или относительных путей к файлам, а также URL (‘http://‘, ‘https://‘, ‘ftp://‘, ‘ftps://‘) в качестве источника импорта.

Примеры импорта:

```
import * from 'set_theory.ontol'
import { limit, derivative } from 'ftp://server.com/calculus.ontol'
import { number as num, integer } from './arithmetic.ontol'
```

3.8. Мета-модель языка

Для целостного представления языка ONTOL V1 на концептуальном уровне была разработана UML-диаграмма, отражающая внутреннюю структуру языка и связи между его компонентами (рис. 2). Эта мета-модель позволяет увидеть язык не только как синтаксическую нотацию, но и как объект для формализации и анализа на метауровне.

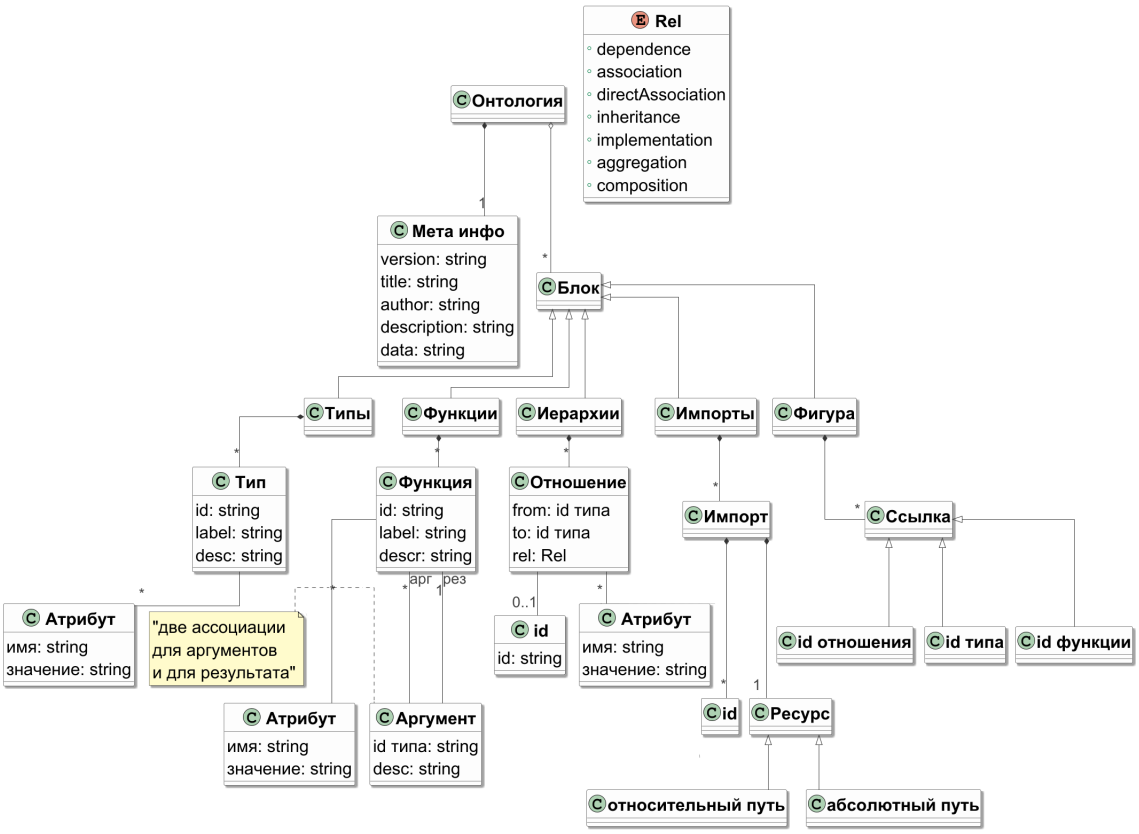

Рис. 2. Мета-модель языка ONTOL V1

Ключевые элементы мета-модели

Блок — корневой контейнер, объединяющий все остальные сущности файла Ontol. Каждый исходный файл трактуется как один *Блок*, содержащий вложенные разделы.

Тип — абстракция предметной сущности. Содержит обязательные атрибуты «id» и «описание» и может иметь визуальные характеристики (цвет, заметка) в качестве атрибутов. Между «Типами» устанавливаются отношения из раздела `Rel`.

Функция — описание процесса или вычисления. Задаёт список входов и выход, что позволяет строить граф потока данных.

Отношение — объект раздела `Иерархии`, фиксирующий семантическую связь между двумя «Типами». Ключевые отношения:

- **dependence (зависимость)** — временная или контекстная связь: один тип «использует» другой внутри операции или процесса.
- **association (ассоциация)** — постоянная, но слабая связь между объектами; они знают друг о друге, но не владеют.
- **directAssociation (прямая ассоциация)** — то же, что association, но без уточняющих ролей; часто применяется для простых связей «A — B».
- **inheritance (наследование)** — отношение «является»; дочерний тип наследует свойства и поведение родительского.
- **implementation (реализация интерфейса)** — дочерний тип обеспечивает контракт (методы/атрибуты), заданный родительским интерфейсом.
- **aggregation (агрегация)** — часть-целое с слабым владением: части могут существовать независимо от целого.
- **composition (композиция)** — часть-целое с сильным владением: «часть» не может существовать без «целого».

Каждому отношению можно задать атрибуты визуализации (цвет стрелки, направление, подпись и т. п.), что позволяет поддерживать читаемость даже при большом числе связей.

Импорт — механизм модульности. Позволяет повторно использовать ранее описанные онтологии, переименовывать сущности через `as` и подключать внешние ресурсы по URL.

Фигура — именованное подмножество модели, формирующее выборку элементов для вывода. Удобно при подготовке учебных схем, когда требуется показать только часть большой онтологии.

Метаданные — набор служебных полей (*version*, *title*, *author*, *description*); их наличие делает файл самодостаточным артефактом с указанием источника, авторства и назначения.

Мета-модель показывает, как в языке отражаются понятия объектно-ориентированного моделирования: например, функции имеют аргументы и возвращаемое значение, а типы могут быть связаны отношениями типа наследования, агрегации и т. п.

Такая визуализация позволяет:

— систематизировать архитектуру языка и его конструкций;
— упростить создание инструментов анализа и трансформации онтологий;
— служить основой для генерации парсеров и визуализаторов;
— использовать модель в учебных целях для объяснения устройства языка студентам.

Мета-модель также способствует дальнейшему развитию языка ONTOL V1 — при добавлении новых конструкций их можно включать в мета-модель, сохраняя целостность и расширяемость системы.

4. ОСНОВНЫЕ ФУНКЦИИ ИНСТРУМЕНТА

- *Непрерывный режим.* Утилита может отслеживать изменения исходного файла и автоматически запускать повторный разбор Ontol-файла, что удобно при итеративной разработке онтологии.
- *Реконструкция исходного кода из AST.* Предусмотрена возможность получения исходной версии Ontol-файла, построенного по разобранному AST[1].
- *Режим без предупреждений.* Для лаконичных отчётов предусмотрена возможность игнорировать все предупреждения и выводить только критически важную информацию.
- *Гибкая маршрутизация вывода.* Результаты работы могут быть сохранены в произвольный каталог.
- *Автогенерация с помощью ИИ (искусственного интеллекта).* Анализатор способен строить иерархические отношения, опираясь на языковые модели из Ollama [10]. Пользователь при этом сам выбирает конкретную модель и задаёт степень «креативности» ответа (температуру), чтобы получить оптимальный баланс точности и новизны.
- *Разделение функций и отношений.* Предусмотрена возможность отображать функции отдельно от отношений между сущностями, что иногда полезно для более чёткого представления; такой режим является стандартной возможностью инструментария.
- *Контроль сложности схем.* Чтобы итоговые диаграммы оставались читаемыми, можно заранее задать предельное число рёбер: анализатор текста автоматически сгенерирует предупреждение (или ошибку), которое будет отображено в отчете.
- *Быстрый просмотр версии.* Вывод текущего номера версии программы осуществляется одной командой, что удобно для скриптов, отслеживающих обновления.
- *Экспорт в JSON.* Ontol CLI (Command Line Interface) поддерживает преобразование онтологии в формат JSON, пригодный для сериализации, хранения и последующего использования во внешних системах. Это особенно важно для интеграции с другими инструментами анализа данных и визуализации.
- *Модульность описаний.* Поддерживается возможность хранения понятий в отдельных Ontol-файлах с последующим импортом используемых. Такой подход способствует повторному использованию, упрощает сопровождение и повышает читаемость крупных онтологий.

Описанные возможности сочетаются: например, можно одновременно отслеживать изменения файла, разделять функции и отношения и сохранять результаты в отдельную папку; при этом игнорировать предупреждения и ограничивать объём финальной схемы. Такая гибкость позволяет настроить Ontol CLI и как интерактивный помощник в IDE, и как часть автоматизированного конвейера.

4.1. Архитектура и жизненный цикл ONTOL CLI

На рисунке 3 показана архитектура CLI-инструмента ONTOL V1.

[1]AST (абстрактное синтаксическое дерево) — структура, отражающая логическую организацию текста Ontol-файла. Используется для визуализации, анализа и трансформации описанной онтологии.

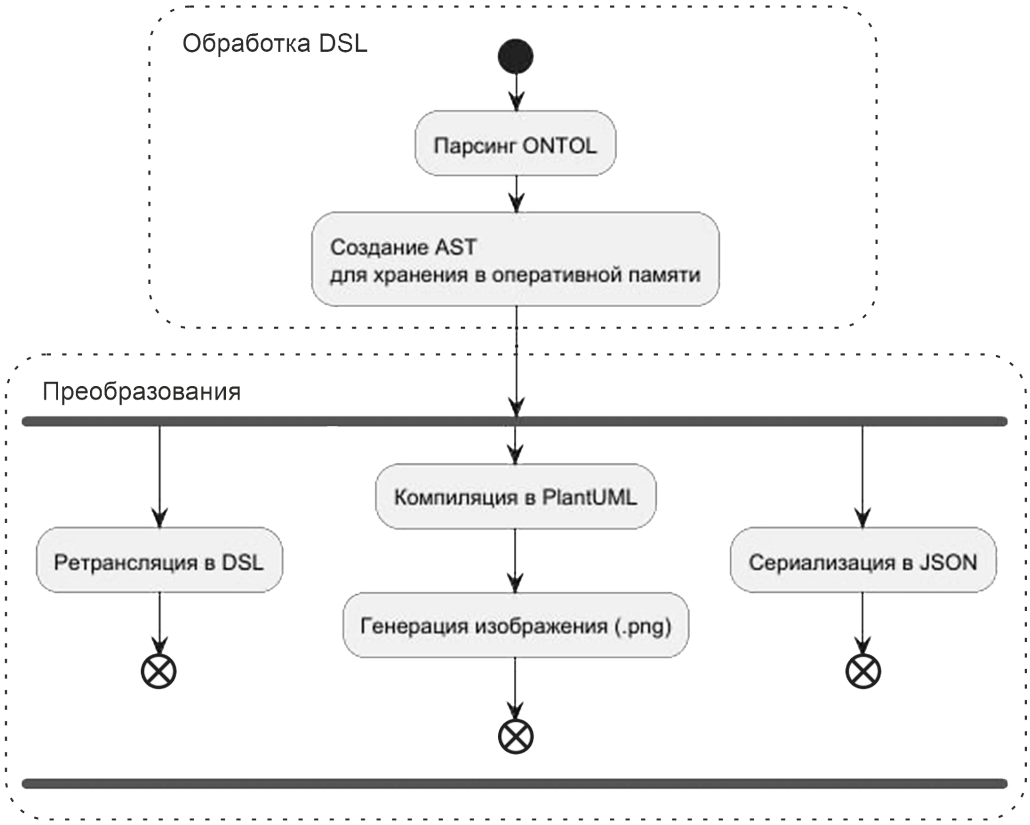

Рис. 3. Диаграмма компонентов и процессов Ontol CLI

В отличие от ручной отрисовки диаграмм в графических редакторах Ontol CLI имеет ряд преимуществ:

— возможность генерации диаграмм программным способом, без ручной вёрстки;
— поддержка модульности и повторного использования — один и тот же элемент может быть включён во множество диаграмм без копирования;
— исправления вносятся централизованно и автоматически распространяются на всю онтологию;
— упрощается сопровождение, развитие и масштабирование описания.

Таким образом, архитектура Ontol CLI демонстрирует не только академическую ценность декларативного описания, но и практическую выгоду при разработке и поддержке крупномасштабных онтологий.

5. ПРИМЕРЫ

В этой секции представлены различные примеры применения языка ONTOL V1, демонстрирующие его возможности в моделировании и описании сложных систем. Эти примеры охватывают широкий спектр областей и включают как специфические, так и общие аспекты, позволяющие увидеть всю мощь и гибкость языка.

5.1. Процессы управления проектами в IT-компании

Для демонстрации возможностей языка ONTOL V1 рассмотрим пример онтологии, изображенной на рисунке 4 и описанной в таблице 1, визуализирующей процессы управления проектами в IT-компании. Онтология охватывает такие сущности, как сотрудники, проекты, задачи и команды, а также связи между ними и ключевые бизнес-функции. Иллюстрации таких диаграмм позволяют глубже понять, как можно использовать онтологии для отображения реальных бизнес-процессов и взаимодействий.

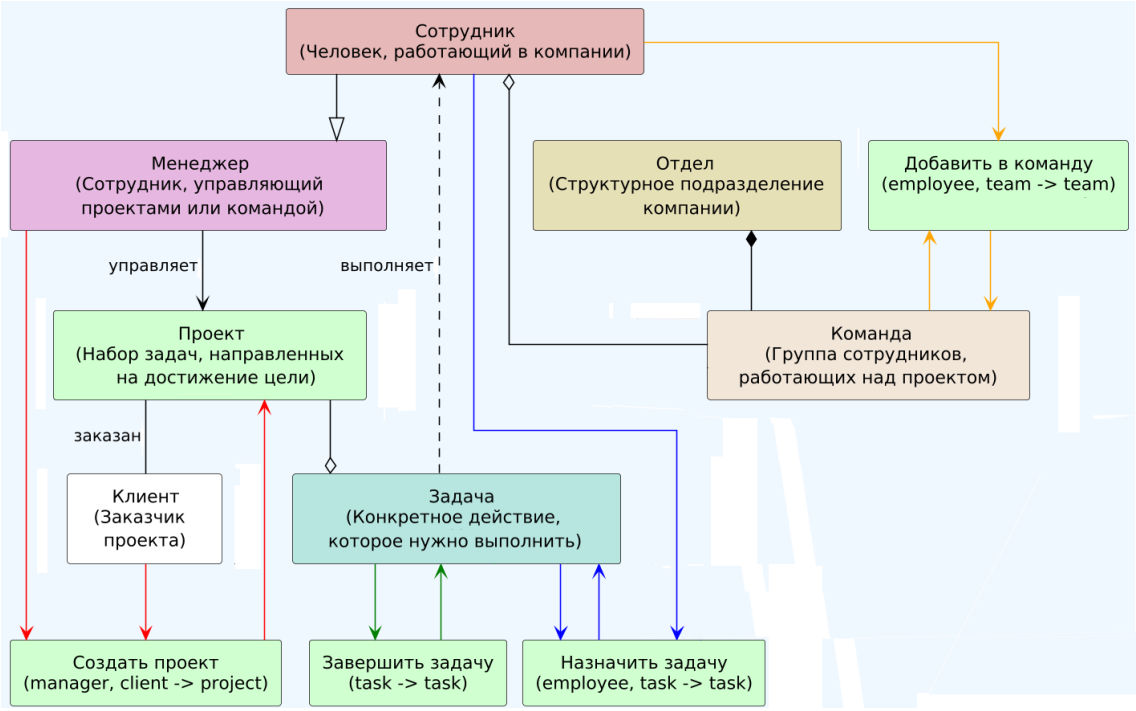

Рис. 4. Диаграмма онтологии управления проектами

Таблица 1. Описание онтологии управления проектами в IT-компании

ONTOL V1	Естественный язык
`version: '1.0', title: 'Управление проектами в IT-компании', author: 'Иван Иванов', description: 'Онтология для описания структуры проектов, сотрудников и задач в IT-компании.'`	Метаданные файла: версия онтологии, её название, автор и её описание.
`types: employee: 'Сотрудник', 'Человек, работающий в компании', { color: '#E6B8B7' }` project: 'Проект', 'Набор задач, направленных на достижение цели', { color: '#D0FFD0' } task: 'Задача', 'Конкретное действие, которое нужно выполнить', { color: '#B7E6E0' }	Типы сущностей: Сотрудник, Проект, Задача, Команда, Отдел, Менеджер и Клиент. Каждый тип описан названием, кратким описанием и цветом для визуализации. Например, Сотрудник — человек, работающий в компании, а Проект — набор задач для дости-

ONTOL V1	Естественный язык
team: 'Команда', 'Группа сотрудников, работающих над проектом', { color: '#F2E6D9' } department: 'Отдел', 'Структурное подразделение компании', { color: '#E6E0B7' } manager: 'Менеджер', 'Сотрудник, управляющий проектами или командой', {color: '#E6B7E0'} client: 'Клиент', 'Заказчик проекта', { color: '#FFFFFF' }	жения целей.
functions: assignTask: 'Назначить задачу' (employee: '', task: '') -> task: '', { color: '#D0FFD0', colorArrow: '#blue' } completeTask: 'Завершить задачу' (task: '') -> task: '', { color: '#D0FFD0', colorArrow: '#green' } createProject: 'Создать проект' (manager: '', client: '') -> project: '', { color: '#D0FFD0', colorArrow: '#red' } addToTeam: 'Добавить в команду' (employee: '', team: '') -> team: '', { color: '#D0FFD0', colorArrow: '#orange' }	Функции: основные действия в онтологии, например, «Назначить задачу» связывает сотрудника с задачей, «Завершить задачу» отмечает её выполненной, «Создать проект» связывает менеджера и клиента с новым проектом, а «Добавить в команду» добавляет сотрудника в конкретную команду.
hierarchy: employee inheritance manager, { direction: 'forward' } team aggregation employee, { direction: 'forward' } project aggregation task, { direction: 'forward' } project association client, { direction: 'bidirectional', title: 'заказан' } manager directAssociation project, { direction: 'forward', title: 'управляет' } task dependence employee, { direction: 'forward', title: 'выполняет' } department composition team, { direction: 'backward' }	Иерархические и ассоциативные связи: Сотрудник наследует от Менеджера, Команда состоит из сотрудников (агрегация), Проект включает задачи, Проект связан с клиентом двусторонней ассоциацией (с названием «заказан»), Менеджер управляет проектом, задачи выполняются сотрудниками, а Отдел включает в себя команды (композиция).

5.2. Множества: зависимости определений

На рисунке 5 представлена UML-диаграмма, иллюстрирующая зависимости между понятиями, связанными с множествами, последовательностями, алфавитами и языками.

Рисунок 5 воспроизводит диаграмму к параграфу 1.1.5 «Алфавит, слово и язык» из [11].

Таким образом, представленный пример иллюстрирует не только академическую структуру математических понятий, но и практическое преимущество перехода от визуального моделирования к декларативному описанию знаний с возможностью автоматической генерации диаграмм.

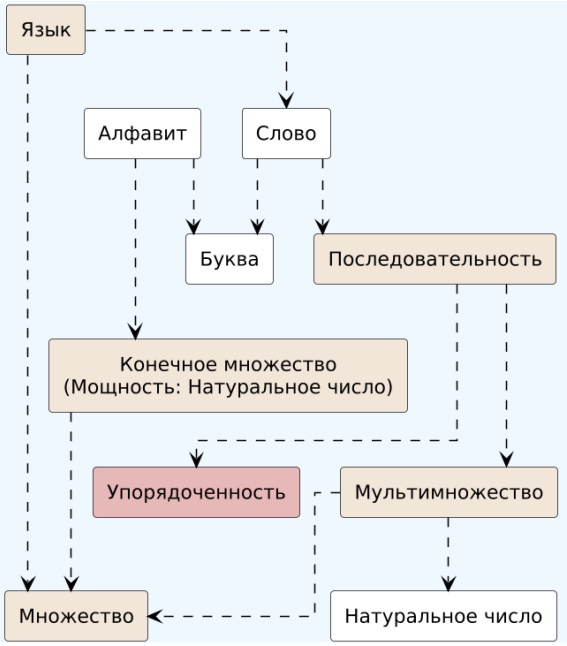

Рис. 5. Диаграмма «Множества — зависимости определений»

6. ЗАКЛЮЧЕНИЕ

В данной работе был представлен язык описания онтологий ONTOL V1 и инструментальное средство командной строки (CLI) для работы с ним. Разработанный язык позволяет формализовать представление онтологий предметных областей в виде текста, сохраняя при этом возможность их наглядной визуализации и обеспечивая машиночитаемость, что является ключевым отличием от чисто графических подходов, таких как UML, и преимуществом в простоте по сравнению с языками вроде OWL, особенно в контексте образовательных задач.

Созданный инструментарий Ontol CLI реализует парсинг языка ONTOL V1, обеспечивает представление онтологии в структурах данных, сериализацию в формат JSON и трансляцию в язык PlantUML для автоматической генерации диаграмм [2], как показано на рисунке 3. Инструмент поддерживает модульность при описании онтологий через систему импортов, гибкую настройку визуализируемых элементов, а также предоставляет вспомогательные режимы для итеративной разработки, такие как отслеживание изменений в файлах (‘–watch‘) и интеграцию нейронных сетей для автоматической генерации отношений (‘–gen-hierarchy‘).

ONTOL V1 предоставляет удобный и формализованный способ для преподавателей и студентов создавать, использовать и обмениваться онтологиями, что способствует систематизации знаний и улучшению восприятия материала, как показал опыт применения в СПбПУ [1]. Полученные машиночитаемые представления онтологий открывают перспективы для дальнейшей автоматизации образовательного процесса, интеграции с системами автоматической проверки знаний и построения интеллектуальных образовательных систем, что продемонстрировано на диаграмме (см. рис. 1). Таким образом, ONTOL V1 вносит вклад в развитие инструментов для представления и использования знаний в инженерном образовании и смежных областях.

Список литературы

1. *Новиков Ф. А., Молотков А. В.* Онтология дискретной математики в образовании // Компьютерные инструменты в образовании. 2021. № 1. С. 5–16.
2. PlantUML Language Reference Guide. [Электронный ресурс]. URL: https://plantuml.com/ (дата обращения: 15.04.2025).
3. *Codd E. F.* A Relational Model of Data for Large Shared Data Banks // Communications of the ACM. 1970. Vol. 13, No. 6. P. 377–387.
4. *Berners-Lee T., Hendler J., Lassila O.* The Semantic Web // Scientific American. 2001. Vol. 284, № 5. P. 34–43.
5. *Klyne G., Carroll J. J.* (Eds.). Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004. [Электронный ресурс]. URL: https://www.w3.org/TR/rdf-concepts/ (дата обращения: 15.04.2025).
6. *Brickley D., Guha R. V.* (Eds.). RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004. [Электронный ресурс]. URL: https://www.w3.org/TR/rdf-schema/ (дата обращения: 15.04.2025).
7. *McGuinness D. L., van Harmelen F.* (Eds.). OWL Web Ontology Language Overview. W3C Recommendation 10 February 2004. [Электронный ресурс]. URL: https://www.w3.org/TR/owl-features/ (дата обращения: 15.04.2025).
8. *Noy N. F., McGuinness D. L.* Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001.
9. *Новиков Ф. А, Иванов Д. Ю.* Моделирование на UML. Теория, практика, видеокурс. СПб.: Профессиональная литература, Наука и Техника. 2010. 6 с.
10. Ollama documentation. [Электронный ресурс]. URL: https://github.com/ollama/ollama/blob/main/docs/api.md (дата обращения: 15.04.2025).
11. *Новиков Ф. А.* Дискретная математика: Учебник для вузов. 2-е изд. Стандарт третьего поколения. СПб.: Питер. 2013.

Поступила в редакцию 10.04.2025, окончательный вариант — 15.04.2025.

Аптуков Михаил Ильдусович, студент программы бакалавриата, высшая школа прикладной математики и вычислительной физики, физико-механический институт, СПбПУ, aptukovm@yandex.ru

Марков Михаил Денисович, студент программы бакалавриата, высшая школа прикладной математики и вычислительной физики, СПбПУ, mike.markov04@gmail.com

Новиков Федор Александрович, доктор техн. наук, профессор, высшая школа прикладной математики и вычислительной физики, физико-механический институт, СПбПУ, fedornovikov51@gmail.com

Пестряков Данил Денисович, аспирант, ассистент, высшая школа прикладной математики и вычислительной физики, физико-механический институт, СПбПУ, ✉ pestryakov.dd@edu.spbstu.ru

Скворцов Владимир Сергеевич, студент программы бакалавриата, высшая школа прикладной математики и вычислительной физики, физико-механический институт, СПбПУ, mail@vladimirskvortsov.com

Хамидуллин Ильсаф Ильназович, студент программы бакалавриата, высшая школа прикладной математики и вычислительной физики, физико-механический институт, СПбПУ, ilsaf.corp@gmail.com

Computer tools in education, 2025
№ 1: 91–107
http://cte.eltech.ru
doi:10.32603/2071-2340-2025-1-91-107

Ontology Description Language ONTOL V1

Aptukov M. I.[1], Student, aptukovm@yandex.ru
Markov M. D.[1], Student, mike.markov04@gmail.com
Novikov F. A.[1], Doctor sc., Professor, fedornovikov51@gmail.com, orcid.org/0000-0003-4450-0173
Pestryakov D. D.[1], Postgraduate, Assistant, ✉ pestryakov.dd@edu.spbstu.ru, orcid.org/0009-0000-6877-5716
Skvortsov V. S.[1], Student, mail@vladimirskvortsov.com, orcid.org/0009-0001-0056-4805
Khamidullin I. I.[1], Student, ilsaf.corp@gmail.com

[1]Peter the Great St. Petersburg Polytechnic University, 29 Polytechnicheskaya, 195251, St. Petersburg, Russia

Abstract

This paper presents the ONTOL V1 ontology description language. With the help of ONTOL V1 notation it is possible to record ontologies applicable in the educational process in a formalised form. In the course of the work, a grammar of the ONTOL V1 language was developed, as well as its parser. The developed software tool provides the user with the following capabilities: recording ontologies in the form of formalised text; machine-readable representation of ontologies; modularity in describing ontologies of subject areas; visualisation of ontology elements.

Keywords: *ontology, subject domain, grammars, education, knowledge formalisation.*

Citation: M. I. Aptukov, M. D. Markov, F. A. Novikov, D. D. Pestryakov, V. S. Skvortsov, I. I. Khamidullin, “Ontology Description Language ONTOL V1,” *Computer tools in education*, no. 1, pp. 91–107, 2025 (in Russian); doi:10.32603/2071-2340-2025-1-91-107

References

1. F. A.Novikov and A. V. Molotkov, “Ontology of Discrete Mathematics in Education,” *Computer Tools in Education*, no. 1, pp. 5–16, 2021 (in Russian).
2. A. Roques, “PlantUML Language Reference Guide,” in *plantuml.com*, 2025. [Online]. Available: https://plantuml.com/
3. E. F. Codd, “A Relational Model of Data for Large Shared Data Banks,”*Communications of the ACM*, vol. 13, no. 6, pp. 377–387, 1970.
4. T. Berners-Lee, J. Hendler, and O. Lassila, “The Semantic Web,” *Scientific American*, vol. 284, no. 5, pp. 34–43, 2001.
5. G. Klyne and J. J. Carroll, eds., “Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation 10 February 2004,” in *w3.org*, 2004. [Online]. Available: https://www.w3.org/TR/rdf-concepts/
6. D. Brickley and R. V. Guha, eds., “RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation 10 February 2004,” in *w3.org*, 2004. [Online]. Available: https://www.w3.org/TR/rdf-schema/

7. D. L. McGuinness and F. van Harmelen, eds., “OWL Web Ontology Language Overview. W3C Recommendation 10 February 2004,” in *w3.org*, 2004. [Online]. Available: https://www.w3.org/TR/owl-features/
8. N. F. Noy and D. L. McGuinness, “Ontology Development 101: A Guide to Creating Your First Ontology,” *Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880*, 2001.
9. F. A. Novikov and D. Yu. Ivanov,*Modeling in UML. Theory, Practice, Video Course*, St. Petersburg, Russia: Professional Literature, Science and Technology, 2010 (in Russian).
10. J. Morgan, “Ollama documentation,” in *github.com*, June 2025. [Online]. Available: https://github.com/ollama/ollama/blob/main/docs/api.md
11. F. A. Novikov, *Discrete Mathematics: Textbook for Universities*, 2nd ed., St. Petersburg, Russia: Peter. 2013 (in Russian).

Received 10-04-2025, the final version — 15-04-2025.

Mikhail Aptukov, Student of the Bachelor’s Degree Program, Department of Applied Mathematics and Informatics, Institute of Physics and Mechanics, SPbPU, aptukovm@yandex.ru

Mikhail Markov, Student of the Bachelor’s Degree Program, Department of Applied Mathematics and Informatics, Institute of Physics and Mechanics, SPbPU, mike.markov04@gmail.com

Fedor Novikov, Doctor of Sciences (Tech.), Professor, Department of Applied Mathematics and Informatics, Institute of Physics and Mechanics, SPbPU, fedornovikov51@gmail.com

Danil Pestryakov, Postgraduate, Assistant, Department of Applied Mathematics and Informatics, Institute of Physics and Mechanics, SPbPU, ✉ pestryakov.dd@edu.spbstu.ru

Vladimir Skvortsov, Student of the Bachelor’s Degree Program, Department of Applied Mathematics and Informatics, Institute of Physics and Mechanics, SPbPU, mail@vladimirskvortsov.com

Ilsaf Khamidullin, Student of the Bachelor’s Degree Program, Department of Applied Mathematics and Informatics, Institute of Physics and Mechanics, SPbPU, ilsaf.corp@gmail.com