УДК 004.82 *Научная статья* DOI: 10.18287/2223-9537-2024-14-1-94-106

Построение базы знаний для автономного управления беспилотными транспортными средствами

© 2024, А.А. Романов, И.А. Рубцов, К.В. Святов, А.А. Филиппов ✉

Ульяновский государственный технический университет, Ульяновск, Россия

Аннотация

Представлен подход к построению и использованию базы знаний для автономного управления беспилотными транспортными средствами. В качестве предметной области представлено сельское хозяйство. Отсутствие достаточного количества и уровня квалификации механизаторов приводит к простою техники, снижению урожайности культур и эффективности использования химикатов. Использование беспилотных транспортных средств позволяет уменьшить влияние данных факторов и снизить вредное воздействие на людей, работающих в сельском хозяйстве. В статье основной упор сделан на учёт особенностей и ограничений предметной области при построении траекторий движения беспилотных транспортных средств и управлении средствами обработки. Предложен подход, состоящий из этапов проектирования схемы базы знаний, автоматизации процесса наполнения базы знаний и организации функции логического вывода. Для каждого этапа приведены разработанные модели и алгоритмы, позволяющие сформировать и использовать базу знаний при решении задачи автономного управления беспилотными транспортными средствами. Статья содержит примеры и иллюстрации, призванные повысить наглядность предложенного подхода.

Ключевые слова: *беспилотные транспортные средства, автономное управление, навигация, сельское хозяйство, база знаний, онтология.*

Цитирование: *Романов А.А., Рубцов И.А., Святов К.В., Филиппов А.А.* Построение базы знаний для автономного управления беспилотными транспортными средствами // Онтология проектирования. 2024. Т.14, №1(51). С.94-106. DOI:10.18287/2223-9537-2024-14-1-94-106.

Финансирование: исследование выполнено за счёт гранта Российского научного фонда (проект № 23-11-00265).

Конфликт интересов: авторы заявляют об отсутствии конфликта интересов.

Введение

Применение интеллектуальных систем (ИС) для автоматизации бизнес-процессов позволяет снизить интеллектуальную нагрузку на специалистов большого числа предметных областей (ПрО). Широкое распространение получили ИС, в основе которых лежат методы машинного обучения (МО) и нейросетевые модели. МО высоко ценится разработчиками ИС из-за широких возможностей адаптации к решению задач при наличии достаточной обучающей выборки. Однако пока не решён вопрос объяснимости результатов, полученных с помощью МО.

В ряде работ рассматриваются вопросы построения ИС на основе методов инженерии знаний. Например, в работах [1-5] разрабатываются базы знаний (БЗ) и средства поддержки их жизнеспособности. В отличие от МО БЗ позволяют формализовать знания об особенностях ПрО в форме, которая одинаково понятна как человеку, так и вычислительной машине. В БЗ знания эксперта представлены в явном виде.

В данной работе рассмотрен подход к построению и использованию БЗ для автономного управления беспилотными транспортными средствами (БПТС) в сельском хозяйстве (СХ).

1 Постановка задачи

Задача управления БПТС является актуальной [6-10]. Применение БПТС в СХ позволяет повысить производительность труда, снизить нагрузку и воздействие вредных веществ на персонал. Можно выделить два основных подхода к управлению БПТС, это использование:

- *набора географических координат*, по которым передвигается БПТС;
- *систем машинного зрения*.

Для применения БПТС в СХ требуется построение гибридной системы, объединяющей оба подхода, а также включающей подсистему управления средствами обработки СХ полей и культур. Поле является объектом с известными границами, и на нём не предполагается оживлённое движение других объектов. Необходимо построить траекторию движения БПТС с учётом особенностей ПрО. Существует вероятность, что на поле может оказаться посторонний объект, поэтому БПТС должно иметь возможность корректировки траектории. Необходимо также управлять средствами обработки для эффективного расходования удобрений и химикатов. Таким образом, требуется решение следующих подзадач:

- оснащение транспортного средства оборудованием для автономного управления;
- формирование БЗ для построения траектории движения БПТС и управления средствами обработки с учётом особенностей ПрО;
- разработка алгоритма построения траектории движения БПТС;
- разработка модели машинного зрения;
- сбор обучающей выборки для тренировки подсистемы машинного зрения для корректирования траектории движения БПТС;
- сборка системы управления БПТС и средствами обработки.

2 Модель БЗ задачи построения траектории движения БПТС

В работе [11] отмечено, что одной из проблем построения эффективного алгоритма управления БПТС является необходимость учёта особенностей ПрО. Можно выделить следующие особенности ПрО СХ:

- *параметры поля*: координаты границы поля, характеристики почвы, показатели урожайности СХ культур, история использования поля, наличие вредителей и др.;
- *особенности обрабатываемых СХ культур*: требования к почве, требования к процессам обработки, требования к удобрениям и химикатам, влияние вредителей и др.;
- *особенности процесса обработки*: траектория обработки поля, периодичность обработки, требования к средствам обработки, требования к БПТС и др.;
- *параметры и особенности БПТС*: габариты, угловая скорость, вес, мощность двигателя, поддерживаемые средства обработки, принцип установки средств обработки и др.;
- *параметры средств обработки*: способ обработки, объём резервуара для химикатов, требования к мощности двигателя БПТС, требования к способу крепления на БПТС и др.

Для формирования БЗ при решении задачи управления БПТС в работах [12-14] предлагается использовать онтологии, основанные на дескрипционной логике (ДЛ) [15]. Это позволяет использовать машины логического вывода для получения новых знаний и проверки логической целостности и непротиворечивости содержимого БЗ с учётом ограничений ПрО. Онтологии позволяют описывать знания в виде продукционных правил. Известны онтологии для ПрО СХ разных уровней [3, 16-19].

Теоретико-множественное представление модели БЗ имеет вид: $O = \langle TBox, ABox, SWRL \rangle$, где $TBox$ – схема БЗ, которая определяет доступное для использования множество классов сущностей ПрО, их свойства, а также отношения между сущностями; $ABox$ – содержимое БЗ; $SWRL$ – множество продукционных правил на языке *SWRL* [20].

Модель БЗ задачи автономного управления БПТС является фрагментом $TBox$ модели БЗ и представляет собой набор аксиом ДЛ, которые определяют модель знаний БЗ с учётом ограничений и особенностей ПрО решаемой задачи. Таблица 1 содержит операторы ДЛ, которые используются для формализации модели БЗ решаемой задачи.

Таблица 1 – Операторы ДЛ и их соответствие операторам языка *OWL*

ДЛ	OWL	Описание
$\top$	*owl:Thing*	Класс верхнего уровня
$\bot$	*owl:Nothing*	Пустой класс
$A \sqsubseteq B$	*A owl:SubClassOf B*	Включение классов (родитель-потомок)
$A \sqcap B \sqsubseteq \bot$	*[A, B] owl:DisjointClasses*	Непересекающиеся классы
$A \equiv B$	*[A, B] owl:equivalentClasses*	Эквивалентные классы
$A \sqcap B$	*A and B*	Пересечение (конъюнкция) классов
$\forall R.A$	*R only A*	Универсальное ограничение
$\exists R.A$	*R some A*	Экзистенциальное ограничение
$\leq nR.A$	*R exactly n A*	Ограничение кардинальности

$TBox$ модели БЗ формируется администратором БЗ вместе с экспертом данной ПрО.

Для удобства классу $owl{:}\,Thing$ добавлено функциональное свойство (ФС) $hasName$ для определения текстового представления имени сущности, которая унаследована остальными классами БЗ: $\top \sqsubseteq \exists hasName.String \sqcap \forall hasName.String \sqcap = 1 hasName.String$.

Основные классы сущностей модели БЗ для описания особенностей ПрО:

$Field \sqsubseteq \top$ – поле, в пределах которого осуществляется автономное управление БПТС;

$Vehicle \sqsubseteq \top$ – БПТС;

$Tool \sqsubseteq \top$ – средства обработки поля, которые могут быть установлены на БПТС;

$Agricultural \sqsubseteq \top$ – СХ культуры, которые могут располагаться на поле;

$Object \sqsubseteq \top$ – объекты, которые могут мешать передвижению БПТС на поле;

$Coordinate \sqsubseteq \top$ – географические координаты.

Все представленные классы являются потомками класса $owl{:}\,Thing$ (наследование), а также определены в качестве непересекающихся:

$$Field \sqcap Vehicle \sqcap Tool \sqcap Agricultural \sqcap Object \sqcap Coordinate \sqsubseteq \bot.$$

ФС класса $Vehicle$ имеют вид:

$$Vehicle \sqsubseteq \exists width.Double \sqcap \forall width.Double \sqcap = 1 width.Double \sqcap$$
$$\sqcap \exists length.Double \sqcap \forall length.Double \sqcap = 1 lengthDouble \sqcap$$
$$\sqcap \exists radius.Double \sqcap \forall radius.Double \sqcap = 1 radius.Double \sqcap$$
$$\sqcap \exists power.Double \sqcap \forall power.Double \sqcap = 1 power.Double,$$

где $width$ и $length$ – ФС для определения габаритов БПТС: ширина и длина соответственно;

$radius$ – ФС для указания значения радиуса разворота БПТС;

$power$ – ФС, значение которого соответствует мощности двигателя БПТС.

ФС класса $Tool$ имеют вид:

$$Tool \sqsubseteq \exists powerRequired.Double \sqcap \forall powerRequired.Double \sqcap$$
$$\sqcap = 1 powerRequired.Double \sqcap$$
$$\sqcap \sqsubseteq \exists toolType.ToolType \sqcap \forall toolType.ToolType \sqcap = 1 toolType.ToolType$$
$$ToolType \equiv \{sprayer, spreader\},$$

где $powerRequired$ – ФС для определения требований средства обработки к мощности двигателя БПТС; $type$ – ФС, указывающее на тип средства обработки. Возможные значения данного ФС заданы в перечислении $ToolType$.

ФС специфичные для определённых классов средств обработки $Tool$ на примере разбрасывателей удобрений ($Spreader$) и полевых опрыскивателей ($Sprayer$) имеют вид:

$$Spreader \sqsubseteq Tool \sqcap \exists diskDiameter.Integer \sqcap \forall diskDiameter.Integer \sqcap$$
$$\sqcap = 1 diskDiameter.Integer;$$
$$Sprayer \sqsubseteq Tool \sqcap \exists tankVolume.Integer \sqcap \forall tankVolume.Integer \sqcap$$
$$\sqcap = 1 tankVolume.Integer,$$
$$Spreader \sqcap Sprayer \sqsubseteq \bot,$$

где $diskDiameter$ – ФС – значение диаметра разбрасывающих удобрения дисков;
$tankVolume$ – ФС, указывающее на объём резервуара для химикатов;
классы $Spreader$ и $Sprayer$ являются наследниками класса $Tool$ и наследуют его свойства $powerRequired$ и $type$. Эти классы объявлены непересекающимися.

Свойства класса $Agricultural$ имеют вид:

$$Agricultural \sqsubseteq \exists toolTypeRequired.ToolType \sqcap \forall toolTypeRequired.ToolType \sqcap$$
$$\sqcap \exists trackRequired.TrackType \sqcap \forall trackRequired.TrackType \sqcap$$
$$\sqcap = 1 trackRequired.TrackType$$
$$TrackType \equiv \{circular, zigzag\},$$

где $toolTypeRequired$ – свойство, определяющее множество средств обработки, которые могут быть применены к данной СХ культуре;
$trackRequired$ – ФС, значение которого указывает на тип траектории, по которой необходимо перемещаться БПТС, при обработке данной СХ культуры;
$TrackType$ – перечисление, содержащее типы траекторий БПТС.

Класс $Coordinate$ имеет следующие ФС:

$$Coordinate \sqsubseteq \exists longitude.Double \sqcap \forall longitude.Double \sqcap = 1 longitude.Double \sqcap$$
$$\sqcap \exists latitude.Double \sqcap \forall latitude.Double \sqcap = 1 latitude.Double,$$

где $longitude$ и $latitude$ – ФС, определяющие множество координат в виде долготы и широты соответственно.

Свойства класса $Object$:

$$Object \sqsubseteq \sqcap \exists objectType.ObjectType \sqcap \forall objectType.ObjectType \sqcap$$
$$\sqcap = 1 objectType.ObjectType \sqcap$$
$$\sqcap \exists hasCoordinate.Coordinate \sqcap \forall hasCoordinate$$
$$ObjectType \equiv \{ravine, saline\},$$

где $objectType$ – ФС, задающее тип объекта значением из перечисления $ObjectType$, например овраги ($ravine$) и солончаки ($saline$);
$hasCoordinate$ – свойство для определения множества координат объекта на поле.

Свойства класса $Field$:

$$Field \sqsubseteq \exists hasBorderCoordinate.Coordinate \sqcap \forall hasBorderCoordinate.Coordinate \sqcap$$
$$\sqcap \exists hasAgricultural.Agricultural \sqcap \forall hasAgricultural.Agricultural \sqcap$$
$$\sqcap = 1 hasAgricultural.Agricultural \sqcap \exists hasObject.Object,$$

где $hasBorderCoordinate$ – свойство для определения множества координат границы поля;
$hasAgricultural$ – ФС для определения СХ культуры, которая находится на поле;
$hasObject$ – свойство, указывающее объекты, которые могут быть расположены на поле.

Модель БЗ может уточняться и дополняться в процессе логического вывода с помощью набора продукционных правил.

3 Разработка БЗ для задачи построения траектории движения БПТС

Определены следующие требования к средствам построения и формирования БЗ для задачи автономного управления БПТС. В частности, необходимо наличие средств для:

- формирования схемы ($TBox$) БЗ для описания особенностей ПрО;
- создания продукционных правил, описывающих правила и закономерности ПрО;

- автоматизации формирования содержимого ($ABox$) БЗ на основе схемы ($TBox$);
- проверки непротиворечивости и целостности знаний, а также получения новых знаний и их интерпретации на основе механизма логического вывода.

3.1 Формирование схемы БЗ и продукционных правил

Для формирования схемы БЗ и продукционных правил на языке *SWRL* использован редактор онтологий *Protégé* [21]. Данный редактор основан на библиотеке *OWL API*, позволяет создавать и редактировать продукции на языке *SWRL* с помощью библиотеки *SWRL API* и может использовать машину логического вывода *Pellet*.

В результате схема БЗ и множество продукций на языке *SWRL* представляются в виде онтологии на языке *OWL*.

3.2 Средства для автоматизации формирования содержимого БЗ

Для автоматизации процесса формирования содержимого ($ABox$) БЗ на основе схемы ($TBox$) разработано приложение, основанное на динамической генерации экранных форм для ввода данных на основе структуры метаданных.

В качестве хранилища разработанного приложения используется графовая система управления базами данных (СУБД) *Neo4j* [22]. Данная СУБД не использует реляционную модель данных и не накладывает ограничения на модель данных, что позволяет загружать в *Neo4j* данные любой структуры в виде графа без предварительной подготовки. Для выполнения запросов к содержимому *Neo4j* необходимо знать структуру метаданных. Основным преимуществом использования СУБД *Neo4j* является поддержка транзакций, что делает возможной коллективную работу по заполнению БЗ несколькими экспертами. *Neo4j* имеет специальный язык запросов *Cypher*, который ориентирован для работы с графом и позволяет создавать эффективные запросы к хранилищу.

В качестве источника для получения метаданных используется схема БЗ ($TBox$). Метаданные позволяют:

- генерировать экранные формы для ввода данных с необходимым набором элементов управления;
- накладывать ограничения на тип и наличие данных для отдельных элементов управления;
- формировать запросы на добавление и извлечение данных из *Neo4j*.

Теоретико-множественное представление метаданных имеет вид: $M = \langle E, A, R, Enum \rangle$,
где E – множество сущностей ПрО (классы БЗ); A – множество свойств сущностей; R – множество отношений, которые связывают сущность и её свойства; $Enum$ – множество перечислений, которые используются в качестве допустимого множества значений отдельных свойств.

Множество сущностей имеет вид: $E = \{E_1, E_2, \dots, E_i, \dots, E_n\}$, где $E_i = \langle type, name \rangle$ – i-я сущность метаданных. Каждая сущность имеет свойство $type$ для определения типа сущности и свойство $name$ для определения имени (представления) сущности в пользовательском интерфейсе. Представление задаётся через аннотацию (элемент языка OWL) схемы БЗ. Например, для класса *Field* – $\langle Field, \text{Поле} \rangle$.

Множество атрибутов сущностей можно записать как: $A = \{A_1, A_2, \dots, A_j, \dots, A_o\}$,
где $A_j = \langle type, name, datatype, required \rangle$ – j-й атрибут метаданных. Атрибут метаданных имеет следующие свойства: $type$ – тип атрибута, $name$ – представление атрибута, $datatype$ – тип данных атрибута, $required$ – признак обязательности заполнения.

Тип данных $datatype$ атрибута может принимать следующие значения: простые типы данных (строка, число, дата, булево); ссылка на другую сущность; коллекция (список).

Например, для свойства $hasBorderCoordinate$ создан атрибут со следующими свойствами: $\langle hasBorderCoordinate, \text{Координаты}, Coordinate[], true\rangle$.

Множество перечислений можно представить в виде следующего выражения:

$$Enum = \{Enum_1, Enum_2, \dots, Enum_k, \dots, Enum_p\},$$

где $Enum_k = \langle type, name, values\rangle$ – k-е перечисление, в котором: $type$ – свойство для определения типа перечисления, $name$ – представление сущности, $values$ – множество значений перечисления. Например, перечисление $TrackType$ представлено как $\langle TrackType, \text{Тип траектории}, \{circular, zigzag\}\rangle$.

Множество отношений R имеет вид: $R = \{R_1, R_2, \dots, R_l, \dots, R_q\}$, где $R_l = \langle E_i, A_j\rangle$ – l-е отношение между сущностью E_i и её атрибутом A_j, например, $\langle Field, hasBorderCoordinate\rangle$.

Для получения метаданных на основе схемы БЗ используется функция вида:

$$F^M: TBox \rightarrow M.$$

Функция F^M реализуется следующим алгоритмом:

1) для каждого перечисления схемы БЗ выполнить преобразование:

$$Enum_i^{TBox} \rightarrow Enum_i^M, \text{ например,}$$

$$TrackType \equiv \{circular, zigzag\} \rightarrow \langle TrackType, \text{Тип траектории}, \{circular, zigzag\}\rangle.$$

2) для каждого класса схемы БЗ выполнить преобразование:

$$Class_i \rightarrow E_i, \text{ например, } Field \rightarrow \langle Field, \text{Поле}\rangle.$$

3) для каждого свойства i-го класса схемы БЗ выполнить преобразование:

$$Property_{ij} \rightarrow \{A_j, R_j\}, \text{ например,}$$

$$hasBorderCoordinate \rightarrow \{\langle hasBorderCoordinate, \text{Координаты}, Coordinate[], true\rangle, \langle Field, hasBorderCoordinate\rangle\}.$$

Тип данных атрибута определяется на основе ограничений и типов данных соответствующего свойства схемы БЗ: если у свойства есть экзистенциальное ограничение (∃) и универсальное ограничение (∀) и свойство не является функциональным, то типом данных атрибута является коллекция, иначе не коллекция.

Итоговый тип данных определяется на основе множества допустимых типов значений из схемы БЗ. Признак обязательности заполнения значения атрибута определяется на основе ограничений схемы БЗ: если у свойства есть универсальное ограничение (∀), то атрибут должен быть обязательно заполнен.

Для автоматизации процесса заполнения БЗ клиентская часть приложения получает метаданные для генерации интерфейса и данные для отображения в нём. Генерация интерфейса осуществляется следующей функцией:

$$F^{GUI}: M \times Neo4j \rightarrow GUI.$$

Например, необходимо ввести данные о некотором экземпляре класса $Field$. В клиентской части приложения используются полученные метаданные класса $Field$ для генерации экранной формы приложения, представленной на рисунке 1. Для каждого атрибута A_j на основе значений его

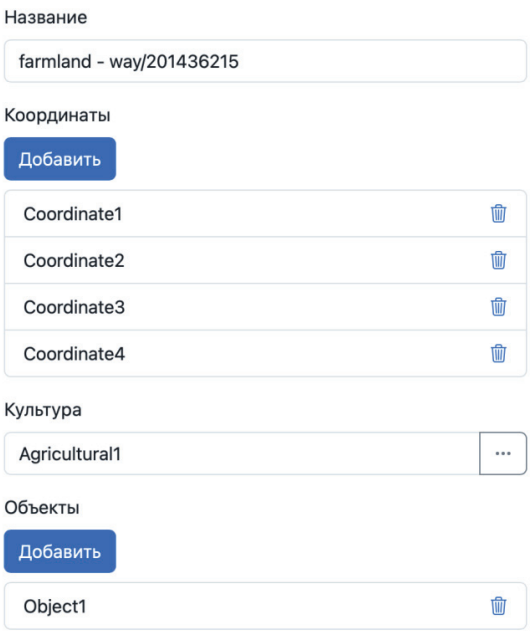


Рисунок 1 – Пример экранной формы для ввода информации о поле

свойств создаётся определённое поле ввода. Например, для атрибута $\langle hasName, \text{Название}, String, true\rangle$ создано следующее поле на языке *HTML*:

```
<div class="mb-3">
    <label for="hasName" class="form-label">Название</label>
    <input type="text" class="form-control" id="hasName" required value="данные">
</div>
```

Как видно из примера разметки, для задания ограничения типа данных используется атрибут *type*, а для указания обязательности заполнения значения атрибут *required* тега *input*. Данные задаются через атрибут *value* тега *input*.

Для атрибута $\langle hasBorderCoordinate, \text{Координаты}, Coordinate[], true\rangle$ *HTML*-разметка имеет вид:

```
<div class="mb-3">
    <label for="hasBorderCoordinate" class="form-label">Координаты</label>
    <br />
    <button class="btn btn-primary" type="button">Добавить</button>
    <div id="hasBorderCoordinate" class="list-group mt-2"
            role="listbox" aria-required="true">
            <div class="list-group-item list-group-item-action d-flex flex-row">
                    <span class="flex-grow-1 align-self-center">данные</span>
                    <a href="#"><i class="bi bi-trash3"></i></a>
            </div>
            …
            <div class="list-group-item list-group-item-action d-flex flex-row">
                    <span class="flex-grow-1 align-self-center">данные</span>
                    <a href="#"><i class="bi bi-trash3"></i></a>
            </div>
    </div>
</div>
```

Так как метаданные содержат все требуемые сведения, при необходимости можно легко заменить шаблоны для генерации элементов управления и представлять коллекции значений в виде таблицы.

Для работы с хранилищем данных *Neo4j* используются следующие функции:

$$F^{StorageInsert}: M \to InsertQuery,\ F^{StorageSelect}: M \to SelectQuery.$$

Функция $F^{StorageInsert}$ позволяет автоматически генерировать запрос на языке *Cypher* для создания записи данных в хранилище *Neo4j*.

Алгоритм работы функции $F^{StorageInsert}$ состоит из следующих шагов.

1) создание узла для добавляемого экземпляра сущности. Например, для объекта класса $Field$: MERGE (e1:**Field**{name: 'Поле'}).
2) поиск атрибутов с типом данных, ссылающихся на экземпляр другой сущности. Например, для класса $Field$ это атрибуты $hasBorderCoordinate$ и $hasObject$. Далее создаются команды для записи в хранилище связанных сущностей. Для атрибута $hasBorderCoordinate$:

```
MERGE (e2:Coordinate{name: 'Координата'})
MERGE (e2a1:Value{value: '54.19'})
MERGE (e2a2:Value{value: '48.22'})
MERGE (e2)-[:latitude]->(e2a1)
MERGE (e2)-[:longitude]->(e2a2).
```

3) создание узлов для атрибутов, которые не ссылаются на другие сущности:

```
MERGE (e1a1:Value{value: 'Поле1'}).
```

4) организация связи между экземпляром сущности и его атрибутами:

```
MERGE (e1)-[:hasName]->(e1a1)
MERGE (e1)-[:hasBorderCoordinate]-(e2).
```

В результате выполнения представленного запроса сформирован фрагмент графа, показанный на рисунке 2.

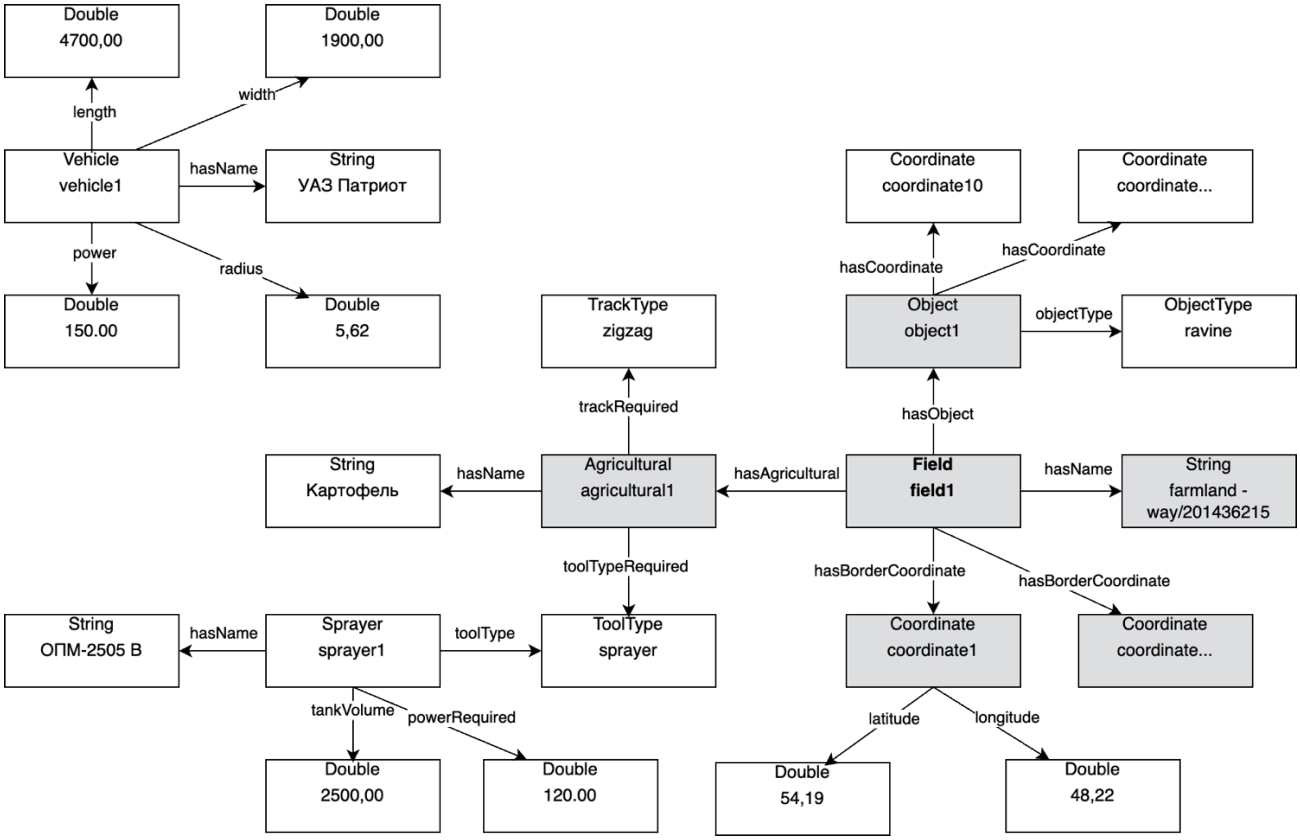


Рисунок 2 – Пример содержимого БЗ

Как видно из рисунка 2, данные о поле записываются в БЗ в качестве фрагмента. Жирным начертанием выделена сущность $\boldsymbol{field1}$ класса $\boldsymbol{Field}$, а фоном отмечены значения свойств сущности $field1$. На рисунке 2 приведены также примеры представления сущностей других классов: БПТС УАЗ Патриот, опрыскиватель полевой штанговый полуприцепной ОПМ-2505 В, СХ культура картофель, овраг, расположенный на поле. Для всех обозначенных сущностей заданы значения их свойств.

Функция $F^{StorageSelect}$ используется для получения данных из хранилища *Neo4j*. Например, для получения списка всех сущностей класса $Field$ используется запрос:

```
MATCH p=(f:Field)-[r*..]->(v) RETURN p.
```

Для получения конкретного поля будет сформирован следующий запрос:

```
MATCH p=(f:Field)-[r*..]->(v) WHERE ID(f)=201 RETURN p.
```

Таким образом, автоматизация формирования содержимого ($ABox$) БЗ снижает нагрузку на эксперта по сравнению с наполнением БЗ средствами редактора *Protégé*.

3.3 Механизм логического вывода

Графовая СУБД *Neo4j* поддерживает механизм логического вывода, но возможности его ограничены. Следовательно, необходимо выполнять логический вывод средствами, разработанными специально для онтологий. Функция логического вывода реализуется с помощью библиотек *OWL API, SWRL API* и машины логического вывода *Pellet*.

Выполнение функций логического вывода позволяет гарантировать корректность содержимого БЗ, логическую целостность и непротиворечивость ограничениям ПрО. В результате

логического вывода на основе множества продукционных правил на языке *SWRL* в БЗ формируются новые знания. Продукционные правила позволяют изменять логику работы с БЗ без изменения алгоритма автономного управления БПТС, что повышает гибкость алгоритма и его живучесть в процессе эксплуатации системы.

Для реализации логического вывода используются следующие функции:

$$F^{Data}: M \times Neo4j \times TBox \rightarrow OWL,$$
$$F^{Inference}: OWL \times Pellet \rightarrow ABox^{*}.$$

Функция F^{Data} на основе метаданных M извлекает из *Neo4j* необходимые данные и записывает их в онтологию в формате *OWL*. При этом заполняется онтология, положенная в основу схемы БЗ $TBox$, полученная на этапе формирования схемы БЗ.

Алгоритм работы функции F^{Data} состоит из следующих шагов.

1) получить множество классов $TBox$ БЗ на основе метаданных M: $E_i \rightarrow Class_i$.
2) для каждой сущности E_i из состава метаданных M выполнить запрос к Neo4j: MERGE (o: E_i)-[:hasName]->(name).
3) создать аксиомы $ABox$ БЗ для определения сущностей для *i*-го класса $Class_i$ БЗ: $name: Class_i$.
4) получить множество свойств *i*-го класса $Class_i$ из модели метаданных M: $\{A_j, R_j\} \rightarrow Property_{ij}$.
5) для каждого *j*-го атрибута A_j сущности E_i из состава метаданных M выполнить запрос к *Neo4j*: MERGE (o: E_i)-[:A_j]->(value).
6) создать аксиомы $ABox$ БЗ для определения *j*-го свойства $Property_{ij}$ сущностей для *i*-го класса $Class_i$ БЗ: $(name, value): Property_{ij}$.

Далее запускается машина логического вывода *Pellet*, которая выполняет логический вывод по содержимому полученной *OWL*-онтологии. Выполнение множества *SWRL*-правил позволяет получать новые знания и использовать их для автономного управления БПТС. В случае возникновения ошибок пользователь должен внести исправления.

Например, *SWRL*-правило, которое позволяет добавить для класса $Field$ дополнительные свойства $fieldTool$ и $fieldTrack$, значения которых указывают на необходимое средство для обработки и тип траектории движения БПТС в зависимости от СХ культуры, имеет вид:

$$Field(?f)\text{^}hasAgricultural(?f,?a)\text{^}toolTypeRequired(?a,?tt)\text{^}toolType(?t,?tt)\text{^}$$
$$\text{^}trackRequired(?a,?tr) \rightarrow fieldTool(?f,?t)\text{^}fieldTrack(?f,?tr).$$

Новые свойства класса $Field$, сформированные в процессе логического вывода, также могут быть использованы в качестве атомов других *SWRL*-правил, например для определения подходящих БПТС на основе требований средств обработки к мощности двигателя:

$$Field(?f)\text{^}fieldTool(?f,?t)\text{^}powerRequired(?t,?pr)\text{^}Vehicle(?v)\text{^}power(?v,?p)\text{^}$$
$$\text{^}swrlb: greaterThanOrEqual(?p,?pr) \rightarrow fieldVehicle(?f,?v).$$

Заключение

В статье рассмотрена задача формирования БЗ для автономного управления БПТС с учётом особенностей ПрО СХ.

Предложены информационная модель БЗ, позволяющая учитывать различные особенности и ограничения при решении задачи автономного управления БПТС, и автоматизированный способ формирования схемы БЗ. Для автоматизации процесса формирования содержимого БЗ предложена информационная модель метаданных, которая используется для генерации динамических экранных форм, позволяющих снизить нагрузку на эксперта и реализовать функцию коллективного заполнения БЗ.

Предложен подход к организации логического вывода для проверки логической целостности и непротиворечивости содержимого БЗ перед выполнением задачи автономного управления БПТС. Механизм логического вывода используется также для получения новых знаний на основе множества продукционных правил на языке *SWRL*, что делает алгоритм построения траектории более гибким.

Список источников

[1] ***Грибова В.В., Москаленко Ф.М., Тимченко В.А., Шалфеева Е.А.*** Создание жизнеспособных интеллектуальных систем с управляемыми декларативными компонентами // Информационные и математические технологии в науке и управлении. 2018. №. 3 (11). С.6-17. DOI: 10.25729/2413-0133-2018-3-01.

[2] ***Кузьмин В.Р., Загорулько Ю.А.*** Применение агентно-сервисного подхода при разработке интеллектуальных систем поддержки принятия решений в энергетике // Вестник Новосибирского государственного университета. Серия: Информационные технологии. 2020. Т.18. №.3. С.5-18. DOI: 10.25205/1818-7900-2020-18-3-5-18.

[3] ***Боргест Н.М., Будаев Д.В., Травин В.В.*** Онтология проектирования точного земледелия: состояние вопроса, пути решения // Онтология проектирования. 2017. Т.7. №4(26). С.423-442. DOI: 10.18287/2223-9537-2017-7-4-423-442.

[4] ***Беглер А.М., Кудрявцев Д.В., Гаврилова Т.А.*** Применение онтологий для интеграции данных эмпирических исследований // Восемнадцатая Национальная конференция по искусственному интеллекту с международным участием КИИ-2020. Труды конференции. Под ред. В.В. Борисова, О.П. Кузнецова. М.: Издательство: МФТИ. 2020. С.3-11.

[5] ***Борисов В.В., Трусов А.С., Кульчицкий П.В., Извозчикова В.В.*** Автоматизация технической диагностики нефтегазового оборудования // Нефтегазовое производство - основа научно-технического прогресса и экономической стабильности. 2020. С.362-367.

[6] ***Xie B., Jin Y., Faheem M., Gao W., Liu J., Jiang H., Cai L., Li Y.*** Research progress of autonomous navigation technology for multi-agricultural scenes // Computers and Electronics in Agriculture. 2023. Vol.211. 107963. DOI: 10.1016/j.compag.2023.107963.

[7] ***Hu J., Gao L., Bai X., Li T., Liu X.*** Review of research on automatic guidance of agricultural vehicles // Transactions of the Chinese Society of Agricultural Engineering. 2015. Vol.31(10). P.1–10.

[8] ***Liu L., Lu S., Zhong R., Wu B., Yao Y., Zhang Q., Shi W.*** Computing systems for autonomous driving: State of the art and challenges // IEEE Internet of Things Journal. 2020. Vol.8(8). P.6469–6486. DOI: 10.1109/JIOT.2020.3043716.

[9] ***Badue C., Guidolini R., Carneiro R.V., Azevedo P., Cardoso V.B., Forechi A., Jesus L., Berriel R., Paixão T.M., Mutz F., de Paula Veronese L., Oliveira-Santos T., De Souza A.F.*** Self-driving cars: A survey // Expert Systems with Applications. 2021. Vol.165. 113816. DOI: 10.1016/j.eswa.2020.113816.

[10] ***Bhalla A., Nikhila M.S., Singh P.*** Simulation of self-driving car using deep learning // 2020 3rd International Conference on Intelligent Sustainable Systems (ICISS). IEEE, 2020. P.519–525. DOI: 10.1109/ICISS49785.2020.9315968.

[11] ***Teeti I., Khan S., Shahbaz A., Bradley A., Cuzzolin F.*** Vision-based Intention and Trajectory Prediction in Autonomous Vehicles: A Survey // Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22, Lud De Raedt, Ed. 2022. Vol.7. P.5630–5637.

[12] ***Chen W., Kloul L.*** An advanced driver assistance test cases generation methodology based on highway traffic situation description ontologies // International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management. Springer, Cham, 2018. P.93–113. DOI: 10.1007/978-3-030-49559-6_5.

[13] ***Ryu M., Cha S.H.*** Context-awareness based driving assistance system for autonomous vehicles // Int. J. Control Autom. 2018. Vol.11(1). P.153–162. DOI: 10.14257/ijca.2018.11.1.14.

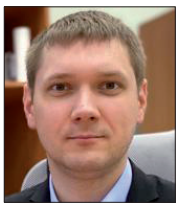
[14] ***Syzdykbayev M., Hajari H., Karimi H.A.*** An ontology for collaborative navigation among autonomous cars, drivers, and pedestrians in smart cities // 4th International Conference on Smart and Sustainable Technologies (SpliTech). IEEE, 2019. P.1–6. DOI: 10.23919/SpliTech.2019.8783045.

[15] The description logic handbook: Theory, implementation and applications / F. Baader (ed.). UK, Cambridge : Cambridge university press, 2003. P.1–23.

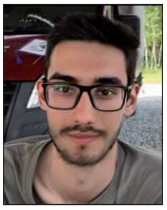
[16] ***Bhuyan B.P., Tomar R., Cherif A.R.*** A Systematic Review of Knowledge Representation Techniques in Smart Agriculture (Urban) // Sustainability. 2022. Vol.14(22). 15249. DOI: 10.3390/su142215249.

[17] ***Saraswathi D., Manibharathy P., Gokulnath R., Sureshkumar E., Karthikeyan K.*** Automation of hydroponics green house farming using IoT // 2018 IEEE International Conference on System, Computation, Automation and Networking (ICSCA). IEEE, 2018. P.1–4. DOI: 10.1109/ICSCAN.2018.8541251.
[18] ***Urkude G., Pandey M.*** AgriOn: a comprehensive ontology for Green IoT based agriculture // Journal Green Eng. 2020. Vol.10(9). P.7078–7101.
[19] ***Jäger M., Nadschläger S., Phan T.N., Küng J.*** Data, information & knowledge sources in the agricultural domain // 2015 26th International Workshop on Database and Expert Systems Applications (DEXA). IEEE, 2015. P.115–119. DOI: 10.1109/DEXA.2015.40.
[20] ***O'Connor M.J., Shankar R.D., Nyulas C., Tu S.W., Das A.K.*** Developing a Web-Based Application using OWL and SWRL // AAAI spring symposium: AI meets business rules and process management. 2008. P.93–98.
[21] Protégé: A free, open-source ontology editor and framework for building intelligent systems: https://protege.stanford.edu.
[22] Neo4j Graph Database & Analytics: https://neo4j.com.

Сведения об авторах



Романов Антон Алексеевич, 1986 г. рождения. Окончил Ульяновский государственный технический университет (УлГТУ) в 2009 г., к.т.н. (2013). Заведующий кафедрой «Информационные системы» УлГТУ. Член Российской ассоциации искусственного интеллекта (РАИИ). В списке научных трудов более 100 работ в области анализа данных и процессов. Author ID (РИНЦ): 684949; Author ID (Scopus): 55903279200. *romanov73@gmail.com.*



Рубцов Иван Алексеевич, 1994 г. рождения. Окончил УлГТУ в 2014 г. Магистр конструкторско-технологического обеспечения машиностроительных производств (2016г). Заведующий лабораторией «Инновационные технологии в машиностроении» УлГТУ. *ria0294@mail.ru*.

Святов Кирилл Валерьевич, 1985 г. рождения. Окончил УлГТУ в 2008 г., к.т.н. (2012). Декан факультета информационных систем и технологий УлГТУ, заведующий кафедрой «Вычислительная техника». В списке научных трудов более 60 работ в области искусственного интеллекта, робототехники и машинного обучения. Author ID (РИНЦ): 609222; Author ID (Scopus): 57209745122. *k.svyatov@gmail.com.*

Филиппов Алексей Александрович, 1987 г. рождения. Окончил УлГТУ в 2009 г., к.т.н. (2013). Доцент кафедры «Информационные системы» УлГТУ. Член Российской ассоциации искусственного интеллекта (РАИИ). В списке научных трудов более 120 работ в области инженерии знаний и анализа данных. Author ID (РИНЦ): 708454; Author ID (Scopus): 57191472723. *al.al.filippov@gmail.com*. ✉

Поступила в редакцию 24.11.2023, после рецензирования 16.01.2024. Принята к публикации 01.02.2024.

Scientific article DOI: 10.18287/2223-9537-2024-14-1-94-106

Building a knowledge base for autonomous control of unmanned vehicles

© 2024, A.A. Romanov, I.A. Rubtsov, K.V. Svyatov, A.A. Filippov ✉

Ulyanovsk State Technical University, Ulyanovsk, Russia

Abstract

This article presents an approach to building and using a knowledge base for autonomous control of unmanned vehicles (UV). Agriculture is presented as a subject area which features and limitations must be considered. The lack of a sufficient number and level of qualifications of machine operator leads to equipment downtime, a decrease in crop yields and the efficiency of using chemical fertilizers. The use of UV makes it possible to reduce the influence of these factors and the harmful effects on people working in agriculture. The article focuses on taking into account the features and limitations of the subject area when constructing the trajectory of unmanned vehicles and controlling processing facilities. An approach is proposed that consists of separate stages of designing a knowledge base schema, automating the process of filling the knowledge base and organizing the logical inference function. For each stage, developed models and algorithms are presented that help to form and use a knowledge base when solving the problem of autonomous control of unmanned vehicles. The article contains examples and illustrations designed to increase the clarity of the proposed approach.

Keywords: *unmanned vehicles, autonomous control, navigation, agriculture, knowledge base, ontology*.

For citation: *Romanov AA, Rubtsov IA, Svyatov KV, Filippov AA*. Building a knowledge base for autonomous control of unmanned vehicles [In Russian]. *Ontology of designing*. 2024; 14(1): 94-106. DOI:10.18287/2223-9537-2024-14-1-94-106.

Financial Support: This work was supported by a grant from the Russian Science Foundation 23-11-00265.

Conflict of interest: The authors declare no conflict of interest.

List of figures and tables

References

[1] ***Gribova VV, Moskalenko FM, Timchenko VA, Shalfeeva EA.*** Building Viable Intelligent Systems with Managed Declarative Components [In Russian]. *Information and mathematical technologies in science and management*. 2018. 3(11). 6-17. DOI: 10.25729/2413-0133-2018-3-01.

[2] ***Kuzmin VR, Zagorulko YA.*** Application of an agent-service approach in the development of intelligent decision support systems in the energy sector [In Russian]. *Bulletin of Novosibirsk State University*. Series: Information technology. 2020. 18(3). 5-18. DOI: 10.25205/1818-7900-2020-18-3-5-18.

[3] ***Borgest NM, Budaev DV, Travin VV.*** Ontology of precision agriculture design: problem state, solution approaches [In Russian]. *Ontology of designing*. 2017. 4(26). 423-442. DOI: 10.18287/2223-9537-2017-7-4-423-442.

[4] ***Begler AM, Kudryavcev DV, Gavrilova TA.*** Using ontologies to integrate empirical research data [In Russian]. *Russian Conference on Artificial Intelligence* (RCAI-2020). 2020. 3-11.

[5] ***Borisov VV, Trusov AS, Kulchitskiy PV, Izvozchikova VV.*** Automation of technical diagnostics of oil and gas equipment [In Russian]. *Oil and gas production is the basis of scientific and technological progress and economic stability*. 2020. 362-367.

[6] ***Xie B, Jin Y, Faheem M, Gao W, Liu J, Jiang H, Cai L, Li Y.*** Research progress of autonomous navigation technology for multi-agricultural scenes. *Computers and Electronics in Agriculture*. 2023; 211: 107963. DOI: 10.1016/j.compag.2023.107963.

[7] ***Hu J, Gao L, Bai X, Li T, Liu X.*** Review of research on automatic guidance of agricultural vehicles. *Transactions of the Chinese Society of Agricultural Engineering*. 2015; 31(10): 1–10.

[8] ***Liu L, Lu S, Zhong R, Wu B, Yao Y, Zhang Q, Shi W.*** Computing systems for autonomous driving: State of the art and challenges. *IEEE Internet of Things Journal*. 2020; 8(8): 6469–6486. DOI: 10.1109/JIOT.2020.3043716.

[9] ***Badue C, Guidolini R, Carneiro RV, Azevedo P, Cardoso VB, Forechi A, Jesus L, Berriel R, Paixão TM, Mutz F, de Paula Veronese L, Oliveira-Santos T, De Souza AF.*** Self-driving cars: A survey. Expert Systems with Applications. 2021; 165: 113816. DOI: 10.1016/j.eswa.2020.113816.

[10] ***Bhalla A, Nikhila MS, Singh P.*** Simulation of self-driving car using deep learning. *2020 3rd International Conference on Intelligent Sustainable Systems* (ICISS). IEEE, 2020. P.519–525. DOI: 10.1109/ICISS49785.2020.9315968.

[11] ***Teeti I, Khan S, Shahbaz A, Bradley A, Cuzzolin F.*** Vision-based Intention and Trajectory Prediction in Autonomous Vehicles: A Survey. *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, IJCAI-22, Lud De Raedt, Ed. 2022; 7: 5630–5637.

[12] ***Chen W, Kloul L.*** An advanced driver assistance test cases generation methodology based on highway traffic situation description ontologies. *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*. Springer, Cham, 2018. P.93–113. DOI: 10.1007/978-3-030-49559-6_5.

[13] ***Ryu M, Cha SH.*** Context-awareness based driving assistance system for autonomous vehicles. *Int. J. Control Autom*. 2018; 11(1): 153–162. DOI: 10.14257/ijca.2018.11.1.14.

[14] ***Syzdykbayev M, Hajari H, Karimi HA.*** An ontology for collaborative navigation among autonomous cars, drivers, and pedestrians in smart cities. *4th International Conference on Smart and Sustainable Technologies* (SpliTech). IEEE, 2019. P.1–6. DOI: 10.23919/SpliTech.2019.8783045.

[15] The description logic handbook: Theory, implementation and applications. F. Baader (ed.). UK, Cambridge: Cambridge university press, 2003. P.1–23.

[16] ***Bhuyan BP, Tomar R, Cherif AR.*** A Systematic Review of Knowledge Representation Techniques in Smart Agriculture (Urban). *Sustainability*. 2022; 14(22): 15249. DOI: 10.3390/su142215249.

[17] ***Saraswathi D, Manibharathy P, Gokulnath R, Sureshkumar E, Karthikeyan K.*** Automation of hydroponics green house farming using IoT. *IEEE International Conference on System, Computation, Automation and Networking* (ICSCA). IEEE, 2018. P.1–4. DOI: 10.1109/ICSCAN.2018.8541251.

[18] ***Urkude G, Pandey M.*** AgriOn: a comprehensive ontology for Green IoT based agriculture. *Journal Green Eng*. 2020; 10(9): 7078–7101.

[19] ***Jäger M, Nadschläger S, Phan TN, Küng J.*** Data, information & knowledge sources in the agricultural domain. *26th International Workshop on Database and Expert Systems Applications* (DEXA). IEEE, 2015. P.115–119. DOI: 10.1109/DEXA.2015.40.

[20] ***O'Connor MJ, Shankar RD, Nyulas C, Tu SW, Das AK.*** Developing a Web-Based Application using OWL and SWRL. *AAAI spring symposium*: AI meets business rules and process management. 2008. P.93–98.

[21] Protégé: A free, open-source ontology editor and framework for building intelligent systems: https://protege.stanford.edu.

[22] Neo4j Graph Database & Analytics: https://neo4j.com.

About the authors

Anton Alekseevich Romanov (b. 1986) graduated from the Ulyanovsk State Technical University (UlSTU) in 2009, PhD (2013). He is the Head of the Department of Information systems at the UlSTU. He is a member of the Russian Association of Artificial Intelligence. He is a co-author of about 100 scientific articles and abstracts in the field of Data and Process Mining. Author ID (RSCI): 684949; Author ID (Scopus): 55903279200. *romanov73@gmail.com.*

Ivan Alekseevich Rubtsov (b. 1994) graduated from the UlSTU in 2014. He received a Master degree in Design and Technological Support of Mechanical Engineering Production in 2016. Head of the Laboratory of innovative technologies in mechanical engineering at the UlSTU. *ria0294@mail.ru.*

Kirill Valerievich Svyatov (b. 1985) graduated from the UlSTU in 2008, PhD (2012). He is the Head of the Department of computer engineering at the UlSTU. He is a co-author of about 60 scientific articles and abstracts in the field of AI, Robotics, and Machine learning. Author ID (RSCI): 609222; Author ID (Scopus): 57209745122. *k.svyatov@gmail.com.*

Aleksey Aleksandrovich Filippov (b. 1987) graduated from the Ulyanovsk State Technical University in 2009, PhD (2013). He is an Associate Professor of the Department of information systems at the UlSTU. He is a member of the Russian Association of Artificial Intelligence. He is a co-author of about 120 scientific articles and abstracts in the field of Knowledge Engineering and Data Mining. Author ID (RSCI): 708454; Author ID (Scopus): 57191472723. *al.al.filippov@gmail.com* ✉.

Received November 24, 2023. Revised January 16, 2024. Accepted February 1, 2024.