mathematics

MDPI

Article

An Approach to Building Decision Support Systems Based on an Ontology Service

Anton Romanov, Julia Stroeva, Aleksey Filippov * and Nadezhda Yarushkina

Department of Information Systems, Ulyanovsk State Technical University, 32 Severny Venetz Street, 432027 Ulyanovsk, Russia; romanov73@gmail.com (A.R.); stroeva95@mail.ru (J.S.); jng@ulstu.ru (N.Y.)
* Correspondence: al.filippov@ulstu.ru; Tel.: +7-908-485-8390

Abstract: Modern decision support systems (DSSs) need components for storing knowledge. Moreover, DSSs must support fuzzy inference to work with uncertainty. Ontologies are designed to represent knowledge of complex structures and to perform inference tasks. Developers must use the OWLAPI and SWRL API libraries to use ontology features. They are impossible to use in DSSs written in programming languages not for Java Virtual Machines. The FuzzyOWL library and the FuzzyDL inference engine are required to work with fuzzy ontologies. The FuzzyOWL library is currently unmaintained and does not have a public Git repository. Thus, it is necessary to develop the ontology service. The ontology service must allow working with ontologies and making fuzzy inferences. The article presents ontology models for decision support, fuzzy inference, and the fuzzy inference algorithm. The article considers examples of DSSs for balancing production capacities and image analysis. The article also describes the architecture of the ontology service. The proposed novel ontology models for decision support make it possible to reduce the time of a knowledge base formation. The ontology service can integrate with external systems with HTTP protocol.

Keywords: decision-making; fuzzy inference; fuzzy ontology; http service

Citation: Romanov, A.; Stroeva, J.; Filippov, A.; Yarushkina, N. An Approach to Building Decision Support Systems Based on an Ontology Service. *Mathematics* **2021**, *9*, 2946. https://doi.org/10.3390/math9222946

Academic Editors: Theodore E. Simos and Charampos Tsitouras

Received: 8 October 2021
Accepted: 16 November 2021
Published: 18 November 2021

1. Introduction

Different organizations or individuals need timely decision support [1–5]. Decision-makers must know the specifics of the context of a subject area for decision support. The context in making decisions determines the conditions and constraints of a subject area [6–8]. The context also describes the characteristics (numerical and non-numerical values) of the analyzed object and its relationship with other entities of a subject area. The context is easier to formalize in terms of qualitative rather than quantitative values [9]. Moreover, the large volume and continuous change of data regarding the analyzed object can prevent timely decision-making.

DSSs allow improving the efficiency of the decision-making process. Decision-makers use DSSs for a complete and objective analysis of subject area objects. DSSs take the values of analyzed object properties and give recommendations to a decision-maker (DM). Moreover, DSSs allow hiding the complexity of the context from DM to focus on managing the object. DM only gets objective information and does not process a large amount of data.

The central part of DSS is an inference system [9–17]. An inference system allows inferring results based on information about an analyzed object within a context. In some cases, knowledge about the analyzed object and context can be represented with a high degree of uncertainty. Then the properties of the analyzed object can be described in linguistic terms. In this case, fuzzy inference systems (FISs) can be used to work with uncertainty.

A FIS can be implemented based on various methods: machine learning [14,16], fuzzy controller or simulator [9–13], knowledge base [6–8,15–17].

A trained machine learning model produces a result in a reasonable amount of time [14,16]. A machine learning model is a black box, so it is difficult to explain the reason for its

decisions. The results of fuzzy controllers or simulators are formed by inference and based on a set of fuzzy rules [9–13]. However, such rules are usually not contextualized, which can produce logical errors and inconsistencies. Knowledge bases require the involvement of an expert and time for development, but they allow describing the analyzed object considering context features [6–8,15–17]. Moreover, knowledge bases allow controlling the logical consistency of facts.

The following main approaches are used for the building of DSS knowledge bases:

- Hybridization of ontological engineering methods with other approaches [16];
- Formation of new inference mechanisms [15];
- Formation of core ontologies to support the DSS building process [15–17].

A significant disadvantage of existing approaches to building a DSS is that it regards focusing on a specific subject area, which makes it hard to develop a DSS for other subject areas.

This article describes a service for creating a DSS based on an ontology with support for fuzzy inference. An ontology forms the basis of a DSS knowledge base. Moreover, the article discusses novel ontology models for organizing decision support processes and fuzzy inference. These ontologies reduce the time to a DSS knowledge base formation. A set of SWRL rules defines the logic of inference in a DSS. The DM must complete the following steps to receive recommendations from the ontology service:

1. Describe the context based on the proposed ontology.
2. Create a set of SWRL-rules for inference.
3. Send data about the analyzed object to the service.

The developed ontology service allows creating a DSS for any subject area.

The ontology service provides the basis for DSS building. The service also allows using the resulting knowledge base within any software application. Now, the OWL API and SWRLAPI libraries are used for full-fledged work with ontologies. These libraries are written for the JVM platform, which does not allow simple usage within other technological stacks. Moreover, the JVM platform has significant hardware requirements. The developed service allows access to inference functions via the HTTP protocol. Thus, the service allows the building of a DSS that can be used within any software system. The proposed ontology models and algorithms make it possible to extend the basis for building a DSS. Any ontology can be used if necessary.

Moreover, the developed service contains a module for fuzzy inference. Currently, there are no easy-to-use/out-of-the-box tools for working with fuzzy ontologies. An ontology model to fuzzy logical inference has been developed. This model allows creating fuzzy sets and describing membership functions and linguistic terms. No additional tools are needed. Crisp numerical values are fuzzified to linguistic terms in the process of fuzzy inference. The obtained degrees of membership are used to calculate the truth degree of the inference result. A user is getting more results (more high recall) within fuzzy inference. Crisp (binary logic) rules and linguistic terms are used for fuzzy inference, the calculation of the truth degree is performed transparently for the user.

The organization of the paper is conducted in the following way. Section 2 describes the basic concepts that the novel developed approach uses. Section 3 describes novel ontology models for DSS creation and fuzzy inference. Section 3 describes the architecture of the developed ontology service. Section 4 presents examples of DSS for balancing production capacities and image analysis created with the ontology service. In Section 5, the approach is discussed in detail. The paper ends with the conclusions.

2. Preliminaries

This section discusses the basic concepts that the proposed approach uses. Section 2.1 contains an introduction to ontologies and descriptive logic. Section 2.2 covers the basics of fuzzy inference.

2.1. Ontologies

Ontological models of knowledge representation allow combining the advantages of declarative and production models [18]. Ontologies can describe the features of a subject area and a set of inference rules. Atoms of logical rules are objects of a subject area.

The following expression can represent any ontology [18]:

$$O = \langle C, I, R, F \rangle,$$

where C is a set of ontology classes. Classes describe entities of a subject area;

I is a set of ontology individuals. Individuals describe instances of ontology classes;

R is a set of ontology properties:

$$R = \{R^C, R^D\},$$

where R^C is a tie between ontology classes (object property); R^D is a tie between ontology class and data type (data property). Ontology ties describe the properties of subject area objects represented by classes. F is a set of ontology interpretation functions. Interpretation functions infer new knowledge based on the knowledge already contained in the ontology.

Ontologies are based on different description logics (DL). DL allow formalizing the description of the subject area. Moreover, DL can guarantee the logical integrity and consistency of the ontology. DL have decidability and relatively low computational complexity. The features of DL provide a compromise between expressiveness and decidability. Moreover, DL differ in terms of expressiveness. The base family of DL is $\mathcal{ALC}$. Extensions of $\mathcal{ALC}$ form new types of logic with a higher level of expressiveness. Any ontology is a set of terminology and assertions in terms of DL [19]:

$$O = \{TBox, ABox\},$$

where $TBox$ is a terminological box. The TBox contains sentences describing concepts and relations between concepts; $ABox$ is a assertional box. The ABox contains ground sentences about relations between individuals and concepts.

The OWL 2 standard is currently used as a formal language for representing ontologies. OWL 2 provides the expressiveness of $\mathcal{SROIQ}(\mathcal{D})$, OWL-DL is based on $\mathcal{SHOIN}(\mathcal{D})$, and for OWL-Lite it is $\mathcal{SHIF}(\mathcal{D})$ [19].

SWRL language and its extension SQWRL [20] are used to describe a set of logical rules in OWL 2.

Table 1 contains DL operators and axioms that can be used to describe terminology and assertions in ontology [19].

Various ontology reasoners are used to inference. The main functions of reasoners are [21]:

- Checking the ontology consistency;
- Formation of classes taxonomy;
- Execution of queries to ontology.

Table 1. DL operators and axioms.

Description	DL	OWL
top (a special class with every individual as an instance)	$\top$	owl: Thing
bottom (an empty class)	$\bot$	owl: Nothing
class inclusion axiom	$A \sqsubseteq B$	A owl: SubClassOf B
disjoint classes axiom	$A \sqcap B \sqsubseteq \bot$	[A, B] owl: DisjointClasses
equivalence classes axiom (or defining classes with necessary and sufficient conditions)	$A \equiv B$	[A, B] owl: equivalentClasses
intersection or conjunction of classes	$A \sqcap B$	A and B
union or disjunction of classes	$A \sqcup B$	A or B
universal restriction axiom	$\forall R.A$	R only A
existential restriction axiom	$\exists R.A$	R some A
cardinality restrictions axiom	$\leq nR.A$	R exactly n A
inverse properties axiom	$Inv(R_1) \sqsubseteq R_2$	$[R_1, R_2]$ owl: InverseProperties
disjoint properties axiom	$Dis(R_1, R_2)$	$[R_1, R_2]$ owl: DisjointProperties
concept assertion axiom (a is an instance of class A)	$a: A$	a: A
role assertion axiom	$(a, b): R$	a R b

2.2. Fuzzy Inference

Fuzzy inference is actively used in various intelligent systems [12–14]. Conclusions about the state of some objects based on the analysis of its current state are formed with fuzzy inference. The following concepts of the fuzzy sets theory are used for fuzzy inference [22,23]:

- Membership functions;
- Linguistic variables;
- Fuzzy implication methods.

A membership function in fuzzy logic determines a membership degree of the elements of a universal set to some fuzzy set.

Fuzzy set for a universal set U and a membership function $\mu: U \to [0, 1]$ is defined as [22,23]:

$$\tilde{A} = \{(u, \mu_A(x)) | u \in U\}.$$

Membership function $\mu_A(x)$ quantitatively grades the membership of the elements of a universal set $u \in U$ to a fuzzy set $\tilde{A}$. Membership degree 0 means that an element is not included in a fuzzy set, and membership degree 1 describes a fully included element. Values between 0 and 1 represent fuzzy included elements.

A linguistic variable in THE fuzzy set theory takes THE semantic of terms in a natural or formal language. Terms represent fuzzy variables and are described by a fuzzy set.

A linguistic variable can be represented as an expression [23]:

$$LT = \{x, T(x), X, G, M\},$$

where x is a variable name; $T(x)$ is a set of values of a linguistic variable x. Each value of a set is a fuzzy variable on a set X; G is a is a set of additional features for generating new x values; M is a mathematical rule for determining the type of membership function for each value formed based on a G set.

For example, for the linguistic variable 'age':

- x is an age;
- X a set of integers from the range [0, 120];
- $T(x)$ is a set of fuzzy variables: 'young', 'mature', 'old'. It is necessary to set a membership function that specifies information about what age should be considered 'young', 'mature', 'old';
- G is a set of additional features: 'very', 'not very'. Additional features are used to create new fuzzy variables. For example, 'very young' and 'not very old', etc.

Fuzzy inference is executed based on fuzzy rules. A set of fuzzy rules form a fuzzy production system in the context of some subject area. A fuzzy rule can be represented as the expression:

$$R = \langle K, E, W \rangle,$$

where K is the core of a fuzzy rule of the following form:

$$K = \langle A \Rightarrow C \rangle,$$

where $A = \{A_1, A_2, \ldots, A_i, \ldots, A_n\}$ is an antecedent of a rule consisting of many atoms A_i; $C = \{C_1, C_2, \ldots, C_i, \ldots, C_k\}$ is a consequent of a rule consisting of atoms C_i. Atoms of a fuzzy rule must represent single and compound statements connected by binary operations AND and OR; E function for determining a truth degree (from range (0; 1)) of a fuzzy rule result; W is a fuzzy rule weight.

The fuzzy inference consists of the following steps:

1. Fuzzification. Fuzzification is used to switch from numerical indicators of object properties to linguistic variables. The values of all input variables are associated with specific values of the membership functions of linguistic terms from antecedents of fuzzy rules at the fuzzification stage. If the truth degrees of all atoms of antecedents of a fuzzy rule are found, then fuzzification is considered fulfilled.
2. Aggregation. A truth degree of antecedents for each rule is determined at the aggregation stage. If an antecedent of a fuzzy rule contains one atom, then a truth degree of an antecedent is a truth degree of this atom. A truth degree of an atom is calculated based on the value of a membership function of a linguistic variable term. If an antecedent of a rule contains several atoms, then a truth degree is calculated based on the truth degrees of the antecedent atoms using fuzzy logic operations. The fuzzy logical AND (*min*) operator is usually used.
3. Activation. A truth degree of each consequent atom of the fuzzy rule is determined at the stage of activation. A truth degree of each consequent atom is equal to the algebraic product of a rule weight and a truth degree of a rule antecedent. If weights of production rules are not specified, then their default values are equal to one. Minimum and average functions can be used to calculate truth degrees in addition to the algebraic product.
4. Accumulation. A membership function is formed for each linguistic variable from the consequent fuzzy rules at the accumulation stage. Accumulation is based on the union of fuzzy sets of all consequent atoms for some linguistic variable.
5. Defuzzification. The result of defuzzification is quantitative (crisp) values for each output linguistic variable based on the results of the accumulation of all output linguistic terms from THE consequences of fuzzy rules.

The considered stages of fuzzy inference can be implemented in various ways. Different bases of fuzzy logic, different approaches to combining sets, different approaches to activation and defuzzification, etc., can be used at different stages. The following fuzzy inference algorithms are actively used [9–14,24]: Mamdani algorithm, Tsukamoto algorithm, Larsen algorithm, Sugeno algorithm, and simplified fuzzy inference algorithm.

3. Novel Ontology Models for Decision Support and Fuzzy Inference

This section presents novel ontology models for decision support and fuzzy inference. Section 3.1 discusses the model and logical representation of the ontology for decision support. Section 3.2 describes the proposed mechanism for THE recommendations inference. Section 3.3 discusses the model and logical representation of the ontology for fuzzy inference. Section 3.4 describes the proposed fuzzy inference algorithm. Section 3.5 presents the architecture of the developed ontology service.

3.1. Model and Logical Representation of the Ontology for Decision Support

The novel model of the ontology for decision support allows to describe:

- Entities for organizing decision support;
- Entities of a subject area.

The model of the ontology for decision support can be presented by the following expression:

$$O^{DM} = \langle Decision, Domain, R^{DM} \rangle, \quad (1)$$

where *Decision* is the component for describing entities for decision support; *Domain* is the component for describing the entities of a subject area; R^{DM} is a set of ties between components.

The *Decision* component looks as follows:

$$Decision = \langle States, Recommendations, Inference, Rules \rangle, \quad (2)$$

where *States* is a set of states of analyzed objects; *Recommendations* is a set of textual recommendations for analyzed objects managing; $Inference$ is a set of inferred states of analyzed objects; *Rules* is a set of interpretation functions represented by SWRL rules.

The *Domain* component looks as follows:

$$Domain = \langle Objects, Entities, R^{Domain} \rangle, \quad (3)$$

where *Objects* is a set of analyzed objects; *Entities* is a set of entities from a subject area that affects on states of analyzed objects; R^{Domain} is a set of ties between the analyzed objects and entities.

R^{DM} can be represented as:

$$R^{DM} = \{R_S^{DM}, R_R^{DM}\},$$

where R_S^{DM} is a set of ties between an analyzed object and its possible states; R_R^{DM} is a set of ties between an analyzed object and recommendations for its management.

Logical representation of the ontology for decision support (Equation (1)) in $\mathcal{SHOIN}(\mathcal{D})$ DL notation looks like:

$$Decision \sqsubseteq \top \quad Domain \sqsubseteq \top \quad Inference \sqsubseteq \top$$
$$Recommendations \sqsubseteq Decision \quad States \sqsubseteq Decision \quad Entities \sqsubseteq Domain$$
$$Objects \sqsubseteq Domain$$
$$Decision \sqcap Domain \sqsubseteq \bot \quad Recommendations \sqcap States \sqsubseteq \bot \quad Entities \sqcap Objects \sqsubseteq \bot$$
$$Decision \sqsubseteq \exists hasDescription.String \sqcap \forall hasDescription.String \sqcap$$
$$\sqcap = 1 hasDescription.String$$
$$Domain \sqsubseteq \exists hasName.String \sqcap \forall hasName.String \sqcap = 1 hasName.String$$
$$Entities \sqsubseteq \forall connectToObject.Objects \sqcap \forall notConnectToObject.Objects$$
$$Objects \sqsubseteq \forall connectToEntity.Entities \sqcap \forall notConnectToEntity.Entities \sqcap$$
$$\sqcap \exists hasRecommendation.Recommendations \sqcap$$
$$\sqcap \forall hasRecommendation.Recommendations \sqcap$$
$$\sqcap \forall hasNumericValue.Double \sqcap = 1 hasNumericValue.Double \sqcap$$
$$\sqcap \exists hasState.State \sqcap \forall hasState.State$$
$$Dis(connectToObject, notConnectToObject)$$
$$Dis(connectToEntity, notConnectToEntity)$$
$$Inv(connectToObject) \sqsubseteq connectToEntity$$
$$Inv(notConnectToObject) \sqsubseteq notConnectToEntity$$

where:

- *Decision* is a parent class for decision support entities (Equation (2)). *Decision* class has the following properties:
  - *hasDescription* is a functional property for defining a textual description;
  - *Decision* and *Domain* classes are disjoint classes. The same individual cannot be an instance of disjoint classes;
- *Domain* is a parent class for describing entities of a subject area (Equation (3)). *Domain* class has the following properties:
  - *hasName* is a functional property for defining the name of an entity;
  - *Decision* and *Domain* classes are disjoint classes;
- $Inference$ is a parent class for inferred states of analyzed objects;
- *Recommendations* is a class for describing recommendations for analyzed objects managing. *Recommendations* class inherits properties from parent class *Decision*. *Recommendations* and *States* classes are disjoint classes;
- *States* is a class for describing the states of analyzed objects. *States* class inherits properties from parent class *Decision*. *Recommendations* and *States* classes are disjoint classes;
- *Entities* is a class for describing entities from a subject area that affects on states of analyzed objects. *Entities* class has the following properties:
  - *Entities* class inherits properties from parent class *Domain*;
  - *Entities* and *Objects* classes are disjoint classes;
  - *connectToObject* is a property for determining ties between individuals of classes *Entities* and *Objects*;
  - *notConnectToObject* is a property for determining non-existence ties between individuals of classes *Entities* and *Objects*;
  - *connectToObject* and *notConnectToObject* are disjoint properties. The same individuals cannot be values of disjoint properties at the same time for the same individual;
  - *connectToObject* property is an inverse property for *connectToEntity* property of class *Objects*. Only one property can be instantiated when using inverse properties. An instance of the second property for two individuals will be inferred automatically;
- *Objects* is a class for describing analyzed objects. The *Objects* class has the following properties:
  - *Objects* class inherits properties from parent class *Domain*;
  - *Objects* and *Entities* classes are disjoint classes;
  - *connectToEntity* is a property for determining ties between individuals of classes *Objects* and *Entities*;
  - *notConnectToEntity* is a property for determining non-existence ties between individuals of classes *Objects* and *Entities*;
  - *hasRecommendation* is a property for defining recommendations for analyzed objects managing;
  - *hasNumericValue* is a functional property for determining a numerical value of a property of analyzed objects;
  - *hasState* is a property for determining the state of analyzed objects;
  - *connectToEntity* and *notConnectToEntity* are disjoint properties;
  - *connectToEntity* property is inverse property for *connectToObject* the property of class *Entities*.

3.2. Proposed Mechanism for Recommendations Inference

This section describes the mechanism for recommendations inference based on the decision support ontology. The mechanism for recommendations inference is based on a set of interpretation functions *Rules* (Equation (2)). Interpretation functions are represented

by logical rules in the SWRL and SQWRL languages. These rules are a significant part of the proposed decision support ontology and are applied all at once.

Let us consider the main types of interpretation functions and examples of their implementation in the SWRL language:

1. Function for a transition from a numerical value of a property of an analyzed object to some state of an object:
   - Calculating a degree of membership of a numerical value in a fuzzy set (described in Section 3.4);
   - Determining the occurrence of a numeric value in a numeric range:

$$Object(?o) \wedge hasNumericValue(?o, ?v) \wedge swrlb{:}greaterThan(?v, 50) \wedge \\ \wedge swrlb{:}lessThanOrEqual(?v, 100) \Rightarrow Inference(?o)$$

2. Function for assigning a textual description to states of an analyzed object:

$$Inference(?o) \wedge State(?s) \Rightarrow hasState(?o, ?s)$$

3. Function for displaying textual descriptions of states of an analyzed object:

$$hasState(?o, ?s) \wedge hasDescription(?s, ?d) \Rightarrow sqwrl{:}select(?o, ?d)$$

4. Function for assigning textual recommendations for objects managing to an analyzed object:

$$hasState(?o1, ?s1) \wedge hasState(?o2, ?s2) \wedge Recommendations(?r) \Rightarrow \\ \Rightarrow hasRecommendation(?o1, ?r) \wedge hasSummarisation(?o2, ?r)$$

5. Function for displaying textual recommendations for analyzed objects managing:

$$hasRecommendation(?o, ?r) \wedge hasDescription(?r, ?d) \Rightarrow \\ \Rightarrow sqwrl{:}selectDistinct(?o, ?r, ?d)$$

3.3. Model and Logical Representation of the Ontology for Fuzzy Inference

This section discusses the model and logical representation of the ontology for fuzzy inference. This ontology allows describing fuzzy sets and defining various membership functions.

The model of the ontology for fuzzy inference can be represented by the following expression:

$$O^{FI} = \langle Sets, Functions, Membership, R^{FI} \rangle, \quad (4)$$

where *Sets* is a set of fuzzy sets; *Functions* is a set of membership functions; *Membership* is a set of membership degrees of entities to fuzzy sets; R^{FI} is a set of ties between components.

At the moment, the ontology service supports the following membership functions:

1. Crisp.
2. Linear.
3. Trapezoidal.
4. Triangular.

Logical representation of the ontology for fuzzy inference (Equation (4)) in $\mathcal{SHOIN}(\mathcal{D})$ DL notation looks like:

$$
\begin{aligned}
&FuzzySet \sqsubseteq \top \quad MembershipFunction \sqsubseteq \top \quad MembershipValue \sqsubseteq \top \quad Term \sqsubseteq \top \\
&Crisp \sqsubseteq MembershipFunction \quad Linear \sqsubseteq MembershipFunction \\
&Trapezoidal \sqsubseteq MembershipFunction \quad Triangular \sqsubseteq MembershipFunction \\
&FuzzySet \sqcap MembershipFunction \sqcap MembershipValue \sqcap Term \sqsubseteq \bot \\
&Crisp \sqcap Linear \sqcap Trapezoidal \sqcap Triangular \sqsubseteq \bot \\
&FuzzySet \sqsubseteq \exists hasName.String \sqcap \forall hasName.String \sqcap = 1 hasName.String \sqcap \\
&\quad \sqcap \forall hasTerm.Term \\
&MembershipFunction \sqsubseteq (Crisp \sqcup Linear \sqcup Trapezoidal \sqcup Triangular) \\
&\quad \sqcap \exists crispValues.Double \sqcap \exists linearValues.Double \sqcap \\
&\quad \sqcap \exists trapezoidalValues.Double \sqcap \exists triangularValues.Double \\
&crispValueMin \sqsubseteq crispValues \quad crispValueMax \sqsubseteq crispValues \\
&linearValueMin \sqsubseteq linearValues \quad linearValueMax \sqsubseteq linearValues \\
&trapezoidalValueMin \sqsubseteq trapezoidalValues \\
&trapezoidalValueMax \sqsubseteq trapezoidalValues \\
&trapezoidalValueMidLow \sqsubseteq trapezoidalValues \\
&trapezoidalValueMidHigh \sqsubseteq trapezoidalValues \\
&triangularValueMin \sqsubseteq triangularValues \\
&triangularValueMid \sqsubseteq triangularValues \\
&triangularValueMax \sqsubseteq triangularValues \\
&Crisp \equiv (\geq 1 crispValueMin \sqcap \geq 1 crispValueMax) \\
&Linear \equiv (\geq 1 linearValueMin \sqcap \geq 1 linearValueMax) \\
&Trapezoidal \equiv (\geq 1 trapezoidalValueMin \sqcap \geq 1 trapezoidalValueMidLow \sqcap \\
&\quad \sqcap \geq 1 trapezoidalValueMidHigh \sqcap \geq 1 trapezoidalValueMax) \\
&Triangular \equiv (\geq 1 triangularValueMin \sqcap \geq 1 triangularValueMid \sqcap \\
&\quad \sqcap \geq 1 triangularValueMax) \\
&MembershipValue \sqsubseteq \exists membership.Double \sqcap \forall membership.Double \sqcap \\
&\quad \sqcap \exists membershipEntity.String \sqcap \forall membershipEntity.String \\
&Term \sqsubseteq \exists membershipFunction.MembershipFunction \sqcap \\
&\quad \sqcap \forall membershipFunction.MembershipFunction \sqcap \\
&\quad \sqcap \exists membershipEntity.String \sqcap \forall membershipEntity.String \\
&\top \sqsubseteq \leq 1 Inv(hasMembershipValue).MembershipValue
\end{aligned}
$$

where:

- *FuzzySet* is a parent class for describing fuzzy sets. *FuzzySet* class has the following properties:
  - *hasName* is a functional property for defining a name of a fuzzy set;
  - *hasTerm* is a property for determining a tie between a fuzzy set and a linguistic term;
  - *FuzzySet*, *MembershipFunction*, *MembershipValue* and *Term* classes are disjoint classes;
- *MembershipFunction* is a parent class for describing a membership function. *MembershipFunction* class has the following properties:
  - *MembershipFunction* class is covered by classes *Crisp*, *Linear*, *Trapezoidal* and *Triangular*;

  - *linearValueMin*, *linearValues*, *trapezoidalValues*, *triangularValues* are properties for determining the parameters of membership functions;
  - *FuzzySet*, *MembershipFunction*, *MembershipValue* and *Term* classes are disjoint classes;
- *Crisp* is a class for describing a crisp membership function. *Crisp* class has the following properties:
  - *crispValueMin*, *crispValueMax* are functional properties for defining the parameters of a crisp membership function. These properties are sub-properties of the *crispValues* property;
  - *Crisp*, *Linear*, *Trapezoidal* and *Triangular* classes are disjoint classes;
- *Linear* is a class for describing a linear membership function. *Linear* class has the following properties:
  - *linearValueMin*, *linearValueMax* are functional properties for defining the parameters of a linear membership function. These properties are sub-properties of the *linearValues* property;
  - *Crisp*, *Linear*, *Trapezoidal* and *Triangular* classes are disjoint classes;
- *Trapezoidal* is a class for describing a trapezoidal membership function. *Trapezoidal* class has the following properties:
  - *trapezoidalValueMin*, *trapezoidalValueMidLow*, *trapezoidalValueMidHigh*, *trapezoidalValueMax* are functional properties for defining the parameters of a trapezoidal membership function. These properties are sub-properties of the *trapezoidalValues* property;
  - *Crisp*, *Linear*, *Trapezoidal* and *Triangular* classes are disjoint classes;
- *Triangular* is a class for describing a triangular membership function. *Triangular* class has the following properties:
  - *triangularValueMin*, *triangularValueMid*, *triangularValueMax* are functional properties for defining the parameters of a triangular membership function. These properties are sub-properties of the *triangularValues* property;
  - *Crisp*, *Linear*, *Trapezoidal* and *Triangular* classes are disjoint classes;
- *MembershipValue* is a class for describing the membership degrees of entities to fuzzy sets. *MembershipValue* class has the following properties:
  - *membership* is a property for determining a value of a membership degree;
  - *membershipEntity* is a property for defining a linguistic term. The name of any entity of the ontology can be used as a term;
  - *FuzzySet*, *MembershipFunction*, *MembershipValue* and *Term* classes are disjoint classes;
- *Term* a class for describing a tie between a linguistic term and a membership function. *Term* class has the following properties:
  - *membershipFunction* is a property for determining a membership function;
  - *membershipEntity* is a property for defining a linguistic term;
  - *FuzzySet*, *MembershipFunction*, *MembershipValue*, and *Term* classes are disjoint classes;
- *hasMembershipValue* is an inverse functional property for determining the degree of membership of any entity ontology in a fuzzy set.

3.4. Fuzzy Inference Algorithm

This section describes the fuzzy inference algorithm. The proposed algorithm implements the following stages of fuzzy inference: fuzzification, aggregation, and activation.

Algorithm of the fuzzification stage of a fuzzy inference:

1. A user specifies a fuzzy set for each numeric indicator. A fuzzy set is defined by the name of an individual of *FuzzySet* class.

2. The degree of membership of a quantitative indicator is calculated for each individual of *Term* class, specified as a range of the property *hasTerm* of selected fuzzy set. The implementation of a membership function is used to calculate a degree of membership, specified as a range of *membershipFunction* property of a current linguistic term.
3. The result of fuzzification is added to the ontology by creating individuals of *MembershipValue* class. *membership* property determines a degree of membership, and a range of *membershipEntity* property is copied from a corresponding individual of *Term* class.

Algorithm of stages of aggregation and activation of fuzzy inference:

1. FT dependencies tree of *Rules* interpretation functions is formed [25].
2. All rules of level with the highest depth index ($i = td(FT)$) are processed: $\tilde{FT} = \{FT_j \in FT | j = i\}$.
3. Following values are calculated for each rule ($\tilde{FT}_j = \langle A \Rightarrow C \rangle$) from set $\tilde{FT}$:
   - truth degree using the intersection of fuzzy sets of atoms of an antecedent of a rule: $A^P = max(A_1, A_2, \ldots, A_n)$. If there is no fuzzy set for an atom, then an atom is skipped;
   - truth degree of a consequent of a rule is set equal to the truth degree of an antecedent condition: $C^P = A^P$. If it is impossible to calculate a truth degree of a consequent, then this step is skipped.
4. Truth degree of consequents and antecedents is calculated, considering calculations performed at the previous step when moving to a higher level of the tree ($i = i - 1$).
5. The algorithm continues to run until it reaches the root node of FT dependency tree ($i = 0$).

The results of SQWRL queries are displayed in descending order of a truth degree of consequents.

3.5. Architecture of the Developed Ontology Service

This section discusses the architecture of the ontology service for building decision support systems. The service is written in the Java language. Moreover, the OWL API library [26] is used to work with OWL ontologies. The SWRLAPI library [27] is used to work with SWRL rules. The service API is based on the REST architectural style [28]. Interaction with the service is based on the HTTP protocol. Data is transmitted in JSON format [29]. PostgreSQL is used as the repository of ontologies. Thus, OWL files are stored in PostgreSQL as binary data.

The service has several types of methods to:

1. working with ontologies:
   - Upload new ontology;
   - Update uploaded ontology;
   - Delete uploaded ontology;
   - Download uploaded ontology;
   - Get a list of all uploaded ontologies.
2. Working with SWRL rules:
   - Get a list of all SWRL rules contained in the specified ontology;
   - Get a list of all SQWRL queries contained in the specified ontology;
   - Execution of a user-selected SQWRL query or queries in a specified ontology (crisp inference);
   - Execution of a user-selected SQWRL query or queries in the specified ontology with the preliminary addition of axioms in the ontology ABox (crisp inference);
   - Execution of a user-selected SQWRL query or queries in a specified ontology (fuzzy inference);
   - Execution of a user-selected SQWRL query or queries in the specified ontology with the preliminary addition of axioms in the ontology ABox (fuzzy inference).

Figure 1 shows the architectural diagram of the developed ontology service.

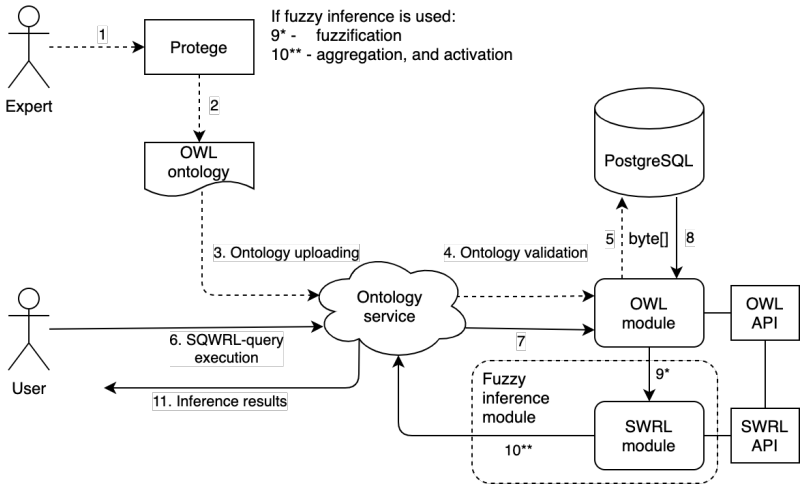

Figure 1. Architectural diagram of the developed ontology service.

Algorithm of interaction with the ontology service from an expert point of view: (Figure 1):

1. Formalization of expert knowledge in the form of ontology.
2. Saving the ontology to an OWL file.
3. Uploading the ontology to the service.
4. Validation of logical integrity and consistency of the ontology.
5. Saving the ontology in PostgreSQL as binary data.

Algorithm of interaction with the ontology service from a user point of view (Figure 1):

6. Execution of SQWRL queries.
7. Selection of a required ontology from a database.
8. Loading a required ontology from PostgreSQL.
9. Fuzzification of numeric values if using fuzzy inference.
10. Aggregation and activation of truth degrees of rules if using fuzzy inference.
11. Output of inference results.

4. Examples of DSSs Created with the Ontology Service

This section presents examples of DSSs created with the ontology service. The extended ontology of decision support (Section 3.1) is used as a knowledge base. The extended fuzzy inference ontology (Section 3.3) and proposed fuzzy inference algorithm (Section 3.4) are used for fuzzy inference.

Moreover, experiments to evaluate the quality of created DSSs are presented in this section. The quality of the DSS is determined using two indicators: precision and recall.

The precision is calculated as:

$$P = |Correct|/|Total| \times 100, \quad (5)$$

where $|Correct|$ is a number of correct DSS results; $|Total|$ is a number of DSS results.

The recall is calculated based on the following expression:

$$R = |Total|/|Rules| \times 100, \quad (6)$$

where $|Total|$ is a number of DSS results; $|Rules|$ is a number of rules assigning textual descriptions and recommendations.

Section 4.1 presents the example of the DSS for balancing production capacities. Section 4.2 describes the example of the DSS for image analysis.

4.1. DSS for Balancing Production Capacities

The capacity management includes the next steps:

- Developing technical passport of the enterprise;
- Calculating the capacities for each production unit and the enterprise as a whole;
- Developing a shortage control strategy;
- Generating a consolidated report with the forecast to implement the product program;
- Calculating the capacity balance.

The balancing of production capacities in this example will be based on the numerical values of the performance indicators of employees (*EmployeePower*) and tools (*ToolPower*).

It is necessary to add the following axioms to the decision support ontology:

- states of production indicators (low, medium, and high) must be added to the the *Decision* component (Equation (2)):

$$High \sqsubseteq Inference \quad Low \sqsubseteq Inference \quad Middle \sqsubseteq Inference$$

- states of production (bad and good) must be added to the the *Decision* component (Equation (2)):

$$Bad \sqsubseteq Inference \quad Good \sqsubseteq Inference$$

- textual representations of states of production indicators must be added to the the *Decision* component (Equation (2)):

$$StateHigh\colon States \quad (StateHigh, \text{“}High\ value\text{”})\colon hasDescription$$
$$StateMiddle\colon States \quad (StateHigh, \text{“}Middle\ value\text{”})\colon hasDescription$$
$$StateLow\colon States \quad (StateHigh, \text{“}Low\ value\text{”})\colon hasDescription$$

- textual representations of recommendations must be added to the the *Decision* component (Equation (2)):

$$EPBad : Recommendations$$
$$(EPBad, \text{“}EPBad\ recommendation\text{”})\colon hasRecommendation$$
$$EPGood : Recommendations$$
$$(EPGood, \text{“}EPGood\ recommendation\text{”})\colon hasRecommendation$$
$$TPBad : Recommendations$$
$$(TPBad, \text{“}TPBad\ recommendation\text{”})\colon hasRecommendation$$
$$TPGood : Recommendations$$
$$(TPGood, \text{“}TPGood\ recommendation\text{”})\colon hasRecommendation$$
$$EPTPBad : Recommendations$$
$$(EPTPBad, \text{“}EPTPBad\ recommendation\text{”})\colon hasRecommendation$$
$$EPTPGood : Recommendations$$
$$(EPTPGood, \text{“}EPTPGood\ recommendation\text{”})\colon hasRecommendation$$

- production indicators and their numerical values must be added to the *Domain* (Equation (3)) component:

$$EmployeePower \sqsubseteq Objects \quad ToolPower \sqsubseteq Objects$$
$$EmployeePower\colon EmployeePower$$
$$(EmployeePower, 4610.41)\colon hasNumericValue$$
$$ToolPower\colon ToolPower$$
$$(ToolPower, 2700.0)\colon hasNumericValue$$

Consider a set of SWRL rules for the transition from numerical values of production indicators to individuals of the *States* class by determining the occurrence of a numeric value in a numeric range:

- *EmployeePower* indicator has state *Low* if its value is in the range of 0 to 2000:

$$EmployeePower(?ind) \wedge hasNumericValue(?ind, ?val) \wedge \\ \wedge swrlb{:}lessThanOrEqual(?val, 2000) \Rightarrow Low(?ind)$$

- *EmployeePower* indicator has state *Middle* if its value is in the range of 2000 to 4000:

$$EmployeePower(?ind) \wedge hasNumericValue(?ind, ?val) \wedge \\ \wedge swrlb{:}greaterThan(?val, 2000) \wedge swrlb{:}lessThanOrEqual(?val, 4000) \Rightarrow \\ \Rightarrow High(?ind)$$

- *EmployeePower* indicator has state *High* if its value is greater than 4000:

$$EmployeePower(?ind) \wedge hasNumericValue(?ind, ?val) \wedge \\ \wedge swrlb{:}greaterThan(?val, 4000) \Rightarrow High(?ind)$$

- *ToolPower* indicator has state *Low* if its value is in the range of 0 to 1000:

$$ToolPower(?ind) \wedge hasNumericValue(?ind, ?val) \wedge \\ \wedge swrlb{:}lessThanOrEqual(?val, 1000) \Rightarrow Low(?ind)$$

- *ToolPower* indicator has state *Middle* if its value is in the range of 1000 to 3000:

$$ToolPower(?ind) \wedge hasNumericValue(?ind, ?val) \wedge \\ \wedge swrlb{:}greaterThan(?val, 1000) \wedge swrlb{:}lessThanOrEqual(?val, 3000) \Rightarrow \\ \Rightarrow High(?ind)$$

- *ToolPower* indicator has state *High* if its value is greater than 3000:

$$ToolPower(?ind) \wedge hasNumericValue(?ind, ?val) \wedge \\ \wedge swrlb{:}greaterThan(?val, 3000) \Rightarrow High(?ind)$$

Next, the following fuzzy sets should be added to the ontology for fuzzy inference:

- for *EmployeePower* indicator: $FuzzEP: FuzzySet$;
- for *ToolPower* indicator: $FuzzTP: FuzzySet$.

It is necessary to add axioms to the ontology for fuzzy inference to determine the membership functions for production indicators for fuzzification of the values of production indicators (Figure 2):

- Linguistic term *Low* for the *EmployeePower* indicator is represented by the following triangular membership function:

$$(TrEPLow, 0): triangularValueMin$$
$$(TrEPLow, 0): triangularValueMid$$
$$(TrEPLow, 2000): triangularValueMax$$
$$TermEPLow: Term$$
$$(TermEPLow, TrEPLow): membershipFunction$$
$$(TermEPLow, \text{“}Low\text{”}): membershipEntity$$
$$(FuzzEP, TermEPLow): hasTerm$$

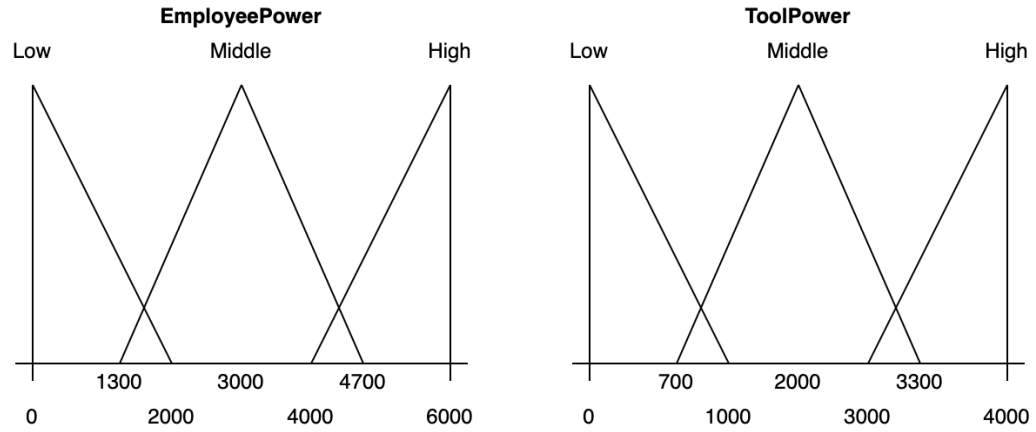

Figure 2. Representation of linguistic variables EmployeePower and ToolPower.

- Linguistic term *Middle* for the *EmployeePower* indicator is represented by the following triangular membership function:

$$(TrEPMiddle, 1300) : triangularValueMin$$
$$(TrEPMiddle, 3000) : triangularValueMid$$
$$(TrEPMiddle, 4700) : triangularValueMax$$
$$TermEPMid : Term$$
$$(TermEPMid, TrEPMiddle) : membershipFunction$$
$$(TermEPMid, \text{“}Middle\text{”}) : membershipEntity$$
$$(FuzzEP, TermEPMid) : hasTerm$$

- Linguistic term *High* for the *EmployeePower* indicator is represented by the following triangular membership function:

$$(TrEPHigh, 4000) : triangularValueMin$$
$$(TrEPHigh, 6000) : triangularValueMid$$
$$(TrEPHigh, 6000) : triangularValueMax$$
$$TermEPHigh : Term$$
$$(TermEPHigh, TrEPHigh) : membershipFunction$$
$$(TermEPHigh, \text{“}High\text{”}) : membershipEntity$$
$$(FuzzEP, TermEPHigh) : hasTerm$$

- Linguistic term *Low* for the *ToolPower* indicator is represented by the following triangular membership function:

$$(TrTPLow, 0) : triangularValueMin$$
$$(TrTPLow, 0) : triangularValueMid$$
$$(TrTPLow, 1000) : triangularValueMax$$
$$TermTPLow : Term$$
$$(TermTPLow, TrTPLow) : membershipFunction$$
$$(TermTPLow, \text{“}Low\text{”}) : membershipEntity$$
$$(FuzzTP, TermTPLow) : hasTerm$$

- Linguistic term *Middle* for the *ToolPower* indicator is represented by the following triangular membership function:

$$(TrTPMiddle, 700) : triangularValueMin$$
$$(TrTPMiddle, 2000) : triangularValueMid$$
$$(TrTPMiddle, 3300) : triangularValueMax$$
$$TermTPMid : Term$$
$$(TermTPMid, TrTPMiddle) : membershipFunction$$
$$(TermTPMid, \text{“}Middle\text{”}) : membershipEntity$$
$$(FuzzTP, TermTPMid) : hasTerm$$

- Linguistic term *High* for the *ToolPower* indicator is represented by the following triangular membership function:

$$(TrTPHigh, 3000) : triangularValueMin$$
$$(TrTPHigh, 4000) : triangularValueMid$$
$$TrTPHigh, 4000) : triangularValueMax$$
$$TermTPHigh : Term$$
$$(TermTPHigh, TrTPHigh) : membershipFunction$$
$$(TermTPHigh, \text{“}High\text{”}) : membershipEntity$$
$$(FuzzTP, TermTPHigh) : hasTerm$$

The following SWRL rules and SQWRL queries are common to the two inference options.

Rules for assigning textual descriptions to states of production indicators:

$$Low(?o) \Rightarrow hasState(?o, StateLow)$$
$$Middle(?o) \Rightarrow hasState(?o, StateMiddle)$$
$$High(?o) \Rightarrow hasState(?o, StateHigh)$$

Rules for determining the state of production:

$$EmployeePower(?ep) \wedge Low(?ep) \Rightarrow Bad(?ep)$$
$$EmployeePower(?ep) \wedge Middle(?ep) \Rightarrow Bad(?ep)$$
$$EmployeePower(?ep) \wedge High(?ep) \Rightarrow Good(?ep)$$
$$ToolPower(?tp) \wedge Low(?tp) \Rightarrow Bad(?tp)$$
$$ToolPower(?tp) \wedge Middle(?tp) \Rightarrow Bad(?tp)$$
$$ToolPower(?tp) \wedge High(?tp) \Rightarrow Good(?tp)$$

Rules for assigning textual recommendations for balancing production capacities:

$$EmployeePower(?ep) \wedge Bad(?ep) \Rightarrow hasRecommendation(?ep, EPBad)$$
$$EmployeePower(?ep) \wedge Good(?ep) \Rightarrow hasRecommendation(?ep, EPGood)$$
$$ToolPower(?tp) \wedge Bad(?tp) \Rightarrow hasRecommendation(?tp, TPBad)$$
$$ToolPower(?tp) \wedge Good(?tp) \Rightarrow hasRecommendation(?tp, TPGood)$$

Complex rules for assigning textual recommendations for balancing production capacities considering several indicators:

$$EmployeePower(?ep) \wedge Bad(?ep) \wedge ToolPower(?tp) \wedge Bad(?tp) \Rightarrow$$
$$\Rightarrow hasRecommendation(?ep, EPTPBad) \wedge hasRecommendation(?tp, EPTPBad)$$
$$EmployeePower(?ep) \wedge Good(?ep) \wedge ToolPower(?tp) \wedge Bad(?tp) \Rightarrow$$
$$\Rightarrow hasRecommendation(?ep, EPTPBad) \wedge hasRecommendation(?tp, EPTPBad)$$
$$EmployeePower(?ep) \wedge Good(?ep) \wedge ToolPower(?tp) \wedge Good(?tp) \Rightarrow$$
$$\Rightarrow hasRecommendation(?tp, EPTPGood) \wedge hasRecommendation(?ep, EPTPGood)$$

The query for displaying text descriptions of the states of production indicators:

$$hasState(?o, ?s) \wedge hasDescription(?s, ?d) \Rightarrow sqwrl{:}select(?o, ?d)$$

The query for displaying textual recommendations for balancing production capacities:

$$hasRecommendation(?o, ?r) \wedge hasDescription(?r, ?d) \Rightarrow sqwrl{:}selectDistinct(?o, ?r, ?d)$$

Results of the query for displaying textual descriptions of production indicator states with crisp inference:

EmployeePower	The indicator value is high
ToolPower	The indicator value is middle

Results of the query for displaying textual descriptions of production indicator states with fuzzy inference:

ToolPower	The indicator value is middle	0.462
EmployeePower	The indicator value is high	0.305
EmployeePower	The indicator value is middle	0.053

Calculated recall (Equation (6)) values for the results of the textual descriptions of production indicator states with crisp and fuzzy inference are:

$$R_1^{Crisp} = 2/10 \times 100\% = 20\%.$$

$$R_1^{Fuzzy} = 3/10 \times 100\% = 30\%.$$

Results of the query for displaying textual recommendations for balancing production capacities with crisp inference:

EmployeePower	EPGood	EPGood recommendation
ToolPower	TPBad	TPBad recommendation
EmployeePower	EPTPBad	EPTPBad recommendation
ToolPower	EPTPBad	EPTPBad recommendation

Results of the query for displaying textual recommendations for balancing production capacities with fuzzy inference:

ToolPower	TPBad	TPBad recommendation	0.462
EmployeePower	EPTPBad	EPTPBad recommendation	0.462
ToolPower	EPTPBad	EPTPBad recommendation	0.462
EmployeePower	EPGood	EPGood recommendation	0.305
EmployeePower	EPBad	EPBad recommendation	0.053

Calculated recall (Equation (6)) values for results of textual recommendations for balancing production capacities with crisp and fuzzy inference are:

$$R_2^{Crisp} = 4/10 \times 100\% = 40\%.$$

$$R_2^{Fuzzy} = 5/10 \times 100\% = 50\%.$$

Thus, the average recall value is higher on 10 % with a fuzzy logical inference, which allows obtaining additional information about the analyzed object.

4.2. DSS for Image Analysis

This section presents an example of the implementation of the DSS for image analysis. This example covers decision support for determining the state of university classrooms. The input data for the system are static images of a classroom. Static images are pre-processed by the YOLO [30] neural network for the entities allocation (Figure 3).

It is necessary to add the following axioms to the decision support ontology:

- Classroom conditions (empty, not empty, and for checking) must be added to the the *Decision* component (Equation (2)):

$$Empty \sqsubseteq Inference \quad NotEmpty \sqsubseteq Inference \quad ForChecking \sqsubseteq Inference$$

- Classrooms must be added to the *Domain* (Equation (3)) component:

$$Classroom \sqsubseteq Objects$$
$$LectureClassroom \sqsubseteq Classroom \; PracticalClassroom \sqsubseteq Classroom$$
$$411 : LectureClassroom \quad (411, \text{“}room\ 411\text{”}) : hasName$$
$$420 : LectureClassroom \quad (420, \text{“}room\ 420\text{”}) : hasName$$
$$431 : PracticalClassroom \quad (431, \text{“}room\ 431\text{”}) : hasName$$

- Entities allocated by YOLO (person and display) must be added to the *Domain* (Equation (3)) component:

$$Display \sqsubseteq Entities \quad Person \sqsubseteq Entities$$
$$Display \sqsubseteq \forall hasAreaValue.Double$$
$$Display : Display \quad Person : Person$$
$$(Display, 22{,}000) : hasSquareValue$$

  hasAreaValue role has been added to store the display area because only situations with a running projector should be handled.
- Then it is necessary to indicate in which classrooms the entities are located:

$$(411, Display) : notConnectToEntity$$
$$(411, Person) : notConnectToEntity$$
$$(420, Display) : notConnectToEntity$$
$$(420, Person) : connectToEntity$$
$$(431, Display) : connectToEntity$$
$$(431, Person) : notConnectToEntity$$

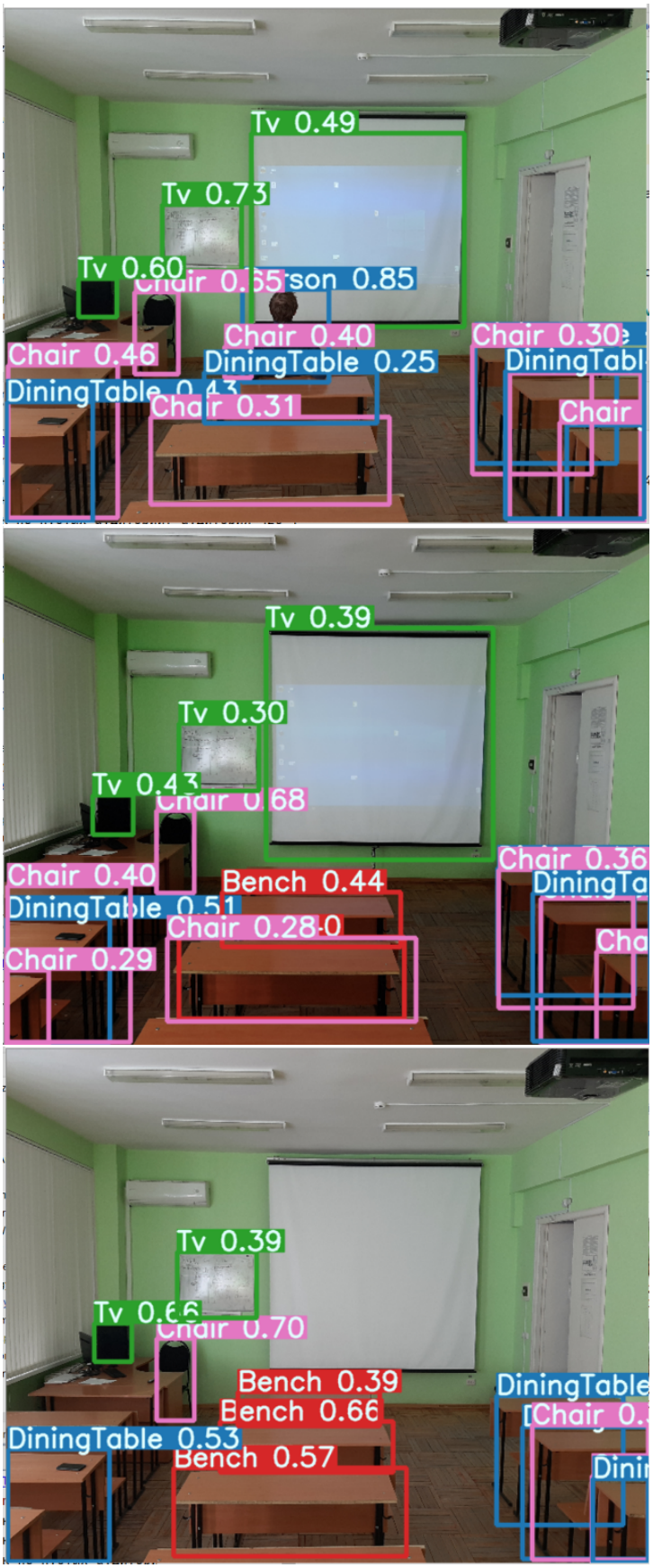

Figure 3. Example of the YOLO neural network results with different scene settings.

Rules for determining the state of classrooms:

$$Classroom(?c) \wedge Person(?p) \wedge notConnectToEntity(?c, ?p) \wedge$$
$$\wedge Display(?d) \wedge notConnectToEntity(?c, ?d) \Rightarrow Empty(?c)$$
$$Classroom(?c) \wedge Person(?p) \wedge connectToEntity(?c, ?p) \Rightarrow NotEmpty(?c)$$
$$Classroom(?c) \wedge Person(?p) \wedge notConnectToEntity(?c, ?p) \wedge$$
$$\wedge Display(?d) \wedge connectToEntity(?c, ?d) \wedge$$
$$\wedge hasAreaValue(?d, ?s) \wedge swrlb{:}greaterThan(?s, 15,000) \Rightarrow ForChecking(?c)$$

The query for displaying empty classrooms:

$$Empty(?c) \wedge hasName(?c, ?n) \wedge abox{:}caa(?t, ?c) \Rightarrow sqwrl : select(?n, ?t)$$

The query for displaying not empty classrooms:

$$NotEmpty(?c) \wedge hasName(?c, ?n) \wedge abox{:}caa(?t, ?c) \Rightarrow sqwrl : select(?n, ?t)$$

The query for displaying classrooms for checking:

$$ForChecking(?c) \wedge hasName(?c, ?n) \wedge abox{:}caa(?t, ?c) \Rightarrow sqwrl : select(?n, ?t)$$

The following states of the classrooms are determined after the execution of these queries:

- 411 is the empty classroom;
- 420 is the not empty classroom;
- 431 is the classroom for checking.

The precision of the DSS was determined by an expert. The expert evaluates the condition of the classrooms based on a review of a set of static images. The number of experiments is 108, the number of correct DSS results is 99. The quality of the DSS was influenced by the YOLO neural network. In some cases, YOLO detected that the projector was on when the projector was turned off.

Thus, the precision value (Equation (5)) is:

$$P = 99/108 \times 100\% = 91\%.$$

5. Discussion

This article discussed an approach to building DSSs based on an ontology service. Knowledge bases allow formalizing the experience of an expert in the decision-making process. The complex process of knowledge base formation is the main disadvantage of knowledge bases. The main advantage of knowledge bases is the ability to explain the reason for the decision. Moreover, the advantage of knowledge bases involves the easy adaptation to changes in a subject area.

The main aim of the study was to create tools for DSS creation. The proposed approach has the following advantages:

- The proposed approach does not depend on a subject area. Existing approaches to build a DSS are focused on a specific subject area, which makes it hard to develop a DSS for other subject areas;
- The developed ontology service allows performing inference functions from any software system using the HTTP protocol. Developers must use the OWLAPI and SWRL API libraries to get all ontology features and abilities. These libraries are written in the Java language, so they cannot be used in systems built on other technologies or systems with a limited set of resources;

- The service API is based on the REST architectural style and the OpenAPI specification. The OpenAPI Specification is a specification for machine-readable interface files for describing, producing, consuming, and visualizing REST API;
- The developed service contains a module for fuzzy inference. The proposed implementation of fuzzy inference is the absence of dependencies on third-party solutions. The proposed ontology for fuzzy inference and the fuzzy inference algorithm implements the fuzzy inference function. Several implementations of fuzzy ontologies are currently available. The most famous implementation of fuzzy ontology is FuzzyOWL [31–33]. Now, the FuzzyOWL library is unmaintained and does not have a public Git repository. Moreover, the plugin for the Protege editor is required to develop fuzzy ontologies for FuzzyOWL. This plugin works only in the 4th version [34] of the Protege editor and does not work in the current 5th version [35].

The proposed approach has the following disadvantages:

- The proposed ontology service only has the unfriendly developer-oriented service API and does not have a user interface;
- The service supports a limited number of membership functions: crisp, linear, trapezoidal, and triangular;
- The proposed approach to the DSS creation cannot be considered production use ready. The tests of the ontology service with a large volume of data and a large number of concurrent requests were not executed;
- The service is based on REST API and does not save request states (stateless). Thus, it is necessary to load an ontology and initialize the inference engine for each request, which requires 2–3 s.
- The ontologies are loaded and processed in RAM. One axiom takes about 0.34 KB of RAM; therefore, 1 GB of RAM can hold about 3 million axioms. Thus, the service is not designed to use large ontologies.

As future work:

- Add support for multi-user mode;
- Add the ability to user register and login;
- Add user ontology repositories (public and private);
- Add a user interface;
- Add the ability to edit and create ontologies through the user interface;
- Publish a service to the public access.

The proposed approach to DSS creation will be improved. The presented ontology service was in its initial state and has been developed for testing purposes. More features and conceptualizations will be added to it.

6. Conclusions

The article discusses an approach to DSS creation based on an ontology service. The article proposes novel ontology models for decision support and fuzzy inference. The ontology of decision support allows describing the features of a subject area and entities for the decision-making. Expert rules are described using SWRL and SQWRL languages. The ontology for fuzzy inference allows describing fuzzy sets, membership functions, and linguistic terms.

The HTTP protocol and REST API are used to work with the ontology service. First, a user needs to create an ontology in the Protege editor. Afterward, a user can upload the created ontology to the service. Decision support is implemented based on an inference mechanism. A user needs to specify which SQWRL query from the ontology needs to be executed to get the result. It is also possible to add axioms to the ontology ABox before executing the SQWRL query. Several alternative solutions can be obtained with varying truth degrees in the case of fuzzy inference.

The article discusses examples of using the proposed approach to create DSSs, for balancing production capacities and for analyzing static images.

Author Contributions: Conceptualization, A.R. and A.F.; methodology, A.R. and A.F.; software, A.R., J.S. and A.F.; validation, A.R., J.S., and A.F.; formal analysis, A.R. and A.F.; investigation, A.R., J.S. and A.F.; resources, J.S.; data curation, A.R. and A.F.; writing—original draft preparation, A.F. and A.R.; writing—review and editing, A.R. and N.Y.; supervision, N.Y.; project administration, N.Y.; funding acquisition, N.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Ministry of Science and Higher Education of the Russian Federation in the framework of the state task no.075-00233-20-05 "Research of intelligent predictive multimodal analysis of big data, and the extraction of knowledge from different sources".

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kukar, M.; Vračar, P.; Košir, D.; Pevec, D.; Bosnić, Z. AgroDSS: A decision support system for agriculture and farming. *Comput. Electron. Agric.* **2019**, *161*, 260–271.
2. Sztubecka, M.; Skiba, M.; Mrówczyńska, M.; Bazan-Krzywoszańska, A. An innovative decision support system to improve the energy efficiency of buildings in urban Areas. *Remote Sens.* **2020**, *12*, 259. [CrossRef]
3. Situmorang, E.; Rindari, F. Decision Support System For Selection Of The Best Doctors In Sari Mutiara Hospital Using Fuzzy Tsukamoto Method. *J. Tek. Inform. CIT Med.* **2019**, *11*, 45–50.
4. Fenu, G.; Malloci, F.M. DSS LANDS: A decision support system for agriculture in Sardinia. *HighTech Innov. J.* **2020**, *1*, 129–135. [CrossRef]
5. Tyler, N.S.; Mosquera-Lopez, C.M.; Wilson, L.M.; Dodier, R.H.; Branigan, D.L.; Gabo, V.B.; Jacobs, P.G. An artificial intelligence decision support system for the management of type 1 diabetes. *Nat. Metab.* **2020**, *2*, 612–619. [CrossRef] [PubMed]
6. Khong, P.C.B.; Lee, L.N.; Dawang, A.I. Modeling the construct of an expert evidence-adaptive knowledge base for a pressure injury clinical decision support system. *Informatics* **2017**, *4*, 20. [CrossRef]
7. Müller, L.; Gangadharaiah, R.; Klein, S.C.; Perry, J.; Bernstein, G.; Nurkse, D.; Aronoff-Spencer, E. An open access medical knowledge base for community driven diagnostic decision support system development. *BMC Med. Inform. Decis. Mak.* **2019**, *19*, 93. [CrossRef]
8. Singh, S.; Ghosh, S.; Jayaram, J.; Tiwari, M.K. Enhancing supply chain resilience using ontology-based decision support system. *Comput. Integr. Manuf.* **2019**, *32*, 642–657. [CrossRef]
9. Vilela, M.; Oluyemi, G.; Petrovski, A. A fuzzy inference system applied to value of information assessment for oil and gas industry. *Decis. Mak. Appl. Manag. Eng.* **2019**, *2*, 1–18. [CrossRef]
10. Precup, R.E.; Preitl, S.; Petriu, E.; Bojan-Dragos, C.A.; Szedlak-Stinean, A.I.; Roman, R.C.; Hedrea, E.L. Model-based fuzzy control results for networked control systems. *Rep. Mech. Eng.* **2020**, *1*, 10–25. [CrossRef]
11. Prasenjit, C. Model for selecting a route for the transport of hazardous materials using a fuzzy logic system. *J. Vojnoteh. Glas.* **2021**, *69*, 355–390.
12. Roy, A.; Razia, S.; Parveen, N.; Rao, A.S.; Nayak, S.R.; Poonia, R.C. Fuzzy rule based intelligent system for user authentication based on user behaviour. *J. Discret. Math. Sci. Cryptogr.* **2020**, *23*, 409–417. [CrossRef]
13. Nagarajan, D.; Lathamaheswari, M.; Jacob, K.; Deenadayalan, E. Intelligent system stability using type-2 fuzzy controller. *Int. J. Integr. Eng.* **2019**, *11*, 270–282. [CrossRef]
14. Espinilla, M.; Medina, J.; García-Fernández, Á.L.; Campaña, S.; Londoño, J. Fuzzy intelligent system for patients with preeclampsia in wearable devices. *Mob. Inf. Syst.* **2017**, *2017*, 7838464. [CrossRef]
15. El-Sappagh, S.; Alonso, J.M.; Ali, F.; Ali, A.; Jang, J.H.; Kwak, K.S. An ontology-based interpretable fuzzy decision support system for diabetes diagnosis. *IEEE Access* **2018**, *6*, 37371–37394. [CrossRef]
16. Alkahtani, M.; Choudhary, A.; De, A.; Harding, J.A. A decision support system based on ontology and data mining to improve design using warranty data. *Comput. Ind. Eng.* **2019**, *128*, 1027–1039. [CrossRef]
17. Mabkhot, M.M.; Al-Samhan, A.M.; Hidri, L. An ontology-enabled case-based reasoning decision support system for manufacturing process selection. *Adv. Mater. Sci. Eng.* **2019**, *2019*, 2505183. [CrossRef]
18. Gruber, T.R. The role of common ontology in achieving sharable, reusable knowledge bases. In Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning, Cambridge, MA, USA, 22–25 April 1991; pp. 601–602.
19. Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; Patel-Schneider, P.F. *The Description Logic Handbook: Theory, Implementation, and Applications*; Cambridge University Press: Cambridge, UK, 2003.
20. O'Connor, M.J.; Das, A.K. SQWRL: A query language for OWL. In Proceedings of the OWLED, Chantilly, VA, USA, 23–24 October 2009.

21. Dentler, K.; Cornet, R.; Ten Teije, A.; De Keizer, N. Comparison of reasoners for large ontologies in the OWL 2 EL profile. *Semant. Web* **2011**, *2*, 71–87. [CrossRef]
22. Zadeh, L.A. Fuzzy logic. *Computer* **1988**, *21*, 83–93. [CrossRef]
23. Zadeh, L.A. The concept of a linguistic variable and its application to approximate reasoning—I. *Inf. Sci.* **1975**, *8*, 199–249. [CrossRef]
24. Ojha, V.; Abraham, A.; Snášel, V. Heuristic design of fuzzy inference systems: A review of three decades of research. *Eng. Appl. Artif. Intell.* **2019**, *85*, 845–864. [CrossRef]
25. Hassanpour, S.; O'Connor, M.J.; Das, A.K. Visualizing Logical Dependencies in SWRL Rule Bases. In *International Workshop on Rules and Rule Markup Languages for the Semantic Web*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 259–272.
26. Horridge, M.; Bechhofer, S. The OWL API: A Java API for Working with OWL 2 Ontologies. In Proceedings of the OWLED, Chantilly, VA, USA, 23–24 October 2009; pp. 11–21.
27. O'Connor, M.J.; Shankar, R.D.; Nyulas, C.; Tu, S.W.; Das, A.K. Developing a Web-Based Application using OWL and SWRL. In Proceedings of the AAAI Spring Symposium: AI Meets Business Rules and Process Management, Stanford, CA, USA, 24–26 March 2008; pp. 93–98.
28. Wilde, E.; Pautasso, C. *REST: From Research to Practice*; Springer Science & Business Media: New York, NY, USA, 2011.
29. Crockford, D. The Application/JSON Media Type for JavaScript Object Notation (JSON). Available online: http://tools.ietf.org/html/rfc4627 (accessed on 10 September 2021).
30. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788.
31. Bobillo, F.; Straccia, U. An OWL ontology for fuzzy OWL 2. In Proceedings of the International Symposium on Methodologies for Intelligent Systems, Prague, Czech Republic, 14–17 September 2009; pp. 151–160.
32. Bobillo, F.; Delgado, M.; Gómez-Romero, J. DeLorean: A reasoner for fuzzy OWL 2. *Expert Syst. Appl.* **2012**, *39*, 258–272. [CrossRef]
33. Fuzzy OWL2 Tools Software & Documentation. Available online: http://www.umbertostraccia.it/cs/software/fuzzyDL/download.html (accessed on 14 September 2021).
34. Protege Wiki. Available online: https://protegewiki.stanford.edu/wiki/FuzzyOWL2 (accessed on 1 September 2021).
35. OWL API Github. Available online: https://github.com/owlcs/owlapi/issues/750 (accessed on 20 September 2021).