ДОДАТОК А

Програмний код

Результат генерації коду онтологічної моделі тексту в Protege

```
import java.util.*;
import null.impl.*;
import null.OntologyJavaMapping;
import edu.stanford.smi.protege.model.*;
import
edu.stanford.smi.protege.code.generator.wrapping.OntologyJavaMapping
Util;

/**
 * Generated by Protege (http://protege.stanford.edu).
 *
*/
public class MyFactory {
    static { OntologyJavaMapping.initMap(); }

    private KnowledgeBase kb;

    public MyFactory(KnowledgeBase kb) {
        this.kb = kb;
    }


    // ***** Class OMTASAT_Class_NP *****

    public Cls getOMTASAT_Class_NPClass() {
        final String name = "OMTASAT_Class_NP";
        return kb.getCls(name);
    }

    public OMTASAT_Class_NP createOMTASAT_Class_NP(String name)
{
        Cls cls = getOMTASAT_Class_NPClass();
        Instance inst = cls.createDirectInstance(name);
        return new DefaultOMTASAT_Class_NP(inst);
    }

    public OMTASAT_Class_NP getOMTASAT_Class_NP(String name) {
        return OntologyJavaMappingUtil.getSpecificObject(kb,
kb.getInstance(name), OMTASAT_Class_NP.class);
    }

    public Set<OMTASAT_Class_NP> getAllOMTASAT_Class_NPObjects()
{
        return getAllOMTASAT_Class_NPObjects(false);
    }

    public Set<OMTASAT_Class_NP>
getAllOMTASAT_Class_NPObjects(boolean transitive) {
        Set<OMTASAT_Class_NP> result = new
HashSet<OMTASAT_Class_NP>();
```

```
            final Cls cls = getOMTASAT_Class_NPClass();
            for (Object element : transitive ? cls.getInstances() :
cls.getDirectInstances()) {
                Instance inst = (Instance) element;

result.add(OntologyJavaMappingUtil.getSpecificObject(kb, inst,
OMTASAT_Class_NP.class));
            }
            return result;
        }


        // ***** Class OMTASAT_Class_POB *****

        public Cls getOMTASAT_Class_POBClass() {
            final String name = "OMTASAT_Class_POB";
            return kb.getCls(name);
        }

        public OMTASAT_Class_POB getOMTASAT_Class_POB(String name) {
            return OntologyJavaMappingUtil.getSpecificObject(kb,
kb.getInstance(name), OMTASAT_Class_POB.class);
        }

        public Set<OMTASAT_Class_POB>
getAllOMTASAT_Class_POBObjects() {
            return getAllOMTASAT_Class_POBObjects(false);
        }

        public Set<OMTASAT_Class_POB>
getAllOMTASAT_Class_POBObjects(boolean transitive) {
            Set<OMTASAT_Class_POB> result = new
HashSet<OMTASAT_Class_POB>();
            final Cls cls = getOMTASAT_Class_POBClass();
            for (Object element : transitive ? cls.getInstances() :
cls.getDirectInstances()) {
                Instance inst = (Instance) element;

result.add(OntologyJavaMappingUtil.getSpecificObject(kb, inst,
OMTASAT_Class_POB.class));
            }
            return result;
        }


        // ***** Class OMTASAT_Class_TD *****

        public Cls getOMTASAT_Class_TDClass() {
            final String name = "OMTASAT_Class_TD";
            return kb.getCls(name);
        }

        public OMTASAT_Class_TD getOMTASAT_Class_TD(String name) {
```

```
            return OntologyJavaMappingUtil.getSpecificObject(kb,
kb.getInstance(name), OMTASAT_Class_TD.class);
        }

        public Set<OMTASAT_Class_TD> getAllOMTASAT_Class_TDObjects()
{
            return getAllOMTASAT_Class_TDObjects(false);
        }

        public Set<OMTASAT_Class_TD>
getAllOMTASAT_Class_TDObjects(boolean transitive) {
            Set<OMTASAT_Class_TD> result = new
HashSet<OMTASAT_Class_TD>();
            final Cls cls = getOMTASAT_Class_TDClass();
            for (Object element : transitive ? cls.getInstances() :
cls.getDirectInstances()) {
                Instance inst = (Instance) element;

result.add(OntologyJavaMappingUtil.getSpecificObject(kb, inst,
OMTASAT_Class_TD.class));
            }
            return result;
        }


        // ***** Class категория_стандарта *****

        public Cls getкатегория_стандартаClass() {
            final String name = "категория_стандарта";
            return kb.getCls(name);
        }

        public категория_стандарта createкатегория_стандарта(String
name) {
            Cls cls = getкатегория_стандартаClass();
            Instance inst = cls.createDirectInstance(name);
            return new Defaultкатегория_стандарта(inst);
        }

        public категория_стандарта getкатегория_стандарта(String
name) {
            return OntologyJavaMappingUtil.getSpecificObject(kb,
kb.getInstance(name), категория_стандарта.class);
        }

        public Set<категория_стандарта>
getAllкатегория_стандартаObjects() {
            return getAllкатегория_стандартаObjects(false);
        }

        public Set<категория_стандарта>
getAllкатегория_стандартаObjects(boolean transitive) {
```

```
            Set<категория_стандарта> result = new
HashSet<категория_стандарта>();
            final Cls cls = getкатегория_стандартаClass();
            for (Object element : transitive ? cls.getInstances() :
cls.getDirectInstances()) {
                Instance inst = (Instance) element;

result.add(OntologyJavaMappingUtil.getSpecificObject(kb, inst,
категория_стандарта.class));
            }
            return result;
        }


        // ***** Class класиф_гр_стандарта *****

        public Cls getкласиф_гр_стандартаClass() {
            final String name = "класиф_гр_стандарта";
            return kb.getCls(name);
        }

        public класиф_гр_стандарта createкласиф_гр_стандарта(String
name) {
            Cls cls = getкласиф_гр_стандартаClass();
            Instance inst = cls.createDirectInstance(name);
            return new Defaultкласиф_гр_стандарта(inst);
        }

        public класиф_гр_стандарта getкласиф_гр_стандарта(String
name) {
            return OntologyJavaMappingUtil.getSpecificObject(kb,
kb.getInstance(name), класиф_гр_стандарта.class);
        }

        public Set<класиф_гр_стандарта>
getAllкласиф_гр_стандартаObjects() {
            return getAllкласиф_гр_стандартаObjects(false);
        }

        public Set<класиф_гр_стандарта>
getAllкласиф_гр_стандартаObjects(boolean transitive) {
            Set<класиф_гр_стандарта> result = new
HashSet<класиф_гр_стандарта>();
            final Cls cls = getкласиф_гр_стандартаClass();
            for (Object element : transitive ? cls.getInstances() :
cls.getDirectInstances()) {
                Instance inst = (Instance) element;

result.add(OntologyJavaMappingUtil.getSpecificObject(kb, inst,
класиф_гр_стандарта.class));
```

ДОДАТОК Б

Слайди презентації

Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Атестаційна робота магістра

Дослідження онтологічних моделей тексту для аналізу технічної документації

Керівник: проф. Лєсна Н.С.
Виконав: ст. гр. ІПЗм-18-2 Вавілов П.О.

1

Об'єкт дослідження

Основна мета полягає у вирішенні завдань, пов'язаних з розробкою технологій формування нормативного профілю для сертифікації програмних систем і інтелектуальної підтримки прийняття рішень аудитором сертифікаційного центру при реалізації основного завдання експертизи програмного забезпечення – формування нормативного профілю.

В роботі методи лінгвістичного аналізу, включаючи і методи здобування знань, мають декларативне подання до вигляді систем продукційних правил.

Застосування продукційних правил забезпечує наступні переваги: простоту і модульність, зручність модифікації, ясність, прозорість, можливість поступового нарощування, високий ступінь спільності правил обробки даних.

2

Онтологія

- це інформаційна структура, що припускає повторне використання і містить комплекс понять, від найзагальніших до найбільш конкретних, що охоплює повний спектр об'єктів і відносин, включаючи події і процеси, а також значення (атрибутів і відносин), що визначаються, якщо необхідно, в часі і просторі .

- Онтологія визначає терміни, використовувані для опису і представлення знань тієї чи іншої предметної області.
- Онтологічна модель знань нормативної бази програмної інженерії – модель предметної області, яка використовує всі доступні засоби представлення знань, релевантні для проведення аналізу галузі формування нормативного профілю.

Застосування онтологій

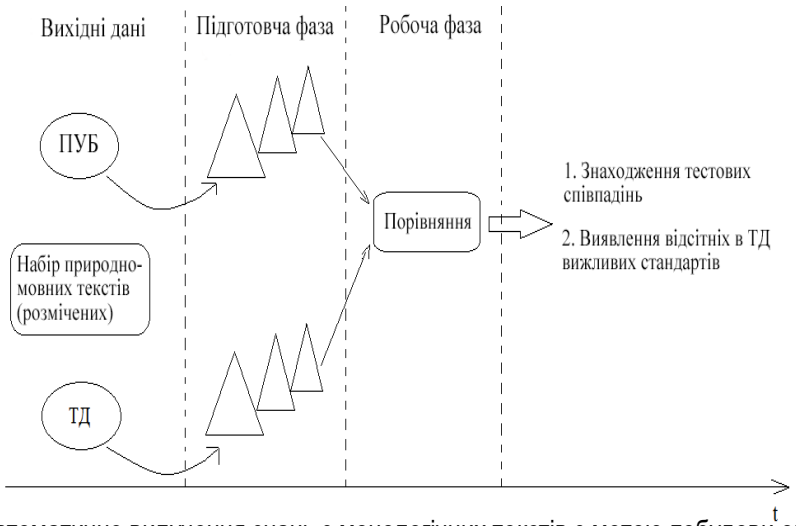
- Онтології можуть бути використані у всіх областях, орієнтованих на застосування методів штучного інтелекту.

- У багатоагентних системах онтології застосовуються як засоби, що надають загальний словник, на підставі яких здійснюється взаємодія агентів, що використовують загальну інтерпретацію знань.

- Ефективне застосування онтологій в області обробки тексту, забезпечується за рахунок операцій, що дозволяють спільне і повторне використання раніше розроблених онтологій.

- Послідовність операцій, орієнтованих на таке використання онтологій, становить **процес управління онтологіями**.

Фази проходження сертифікації



- Автоматичне вилучення знань з монологічних текстів з метою побудови онтології передбачає не тільки виявлення термінів, а й і здобуття знань про терміни.
- Це означає, що для опису семантичної структури термінології необхідно розпізнати в тексті, як терміни, так і семантичні відносини між термінами.

5

Завдання семантичного аналізу тексту

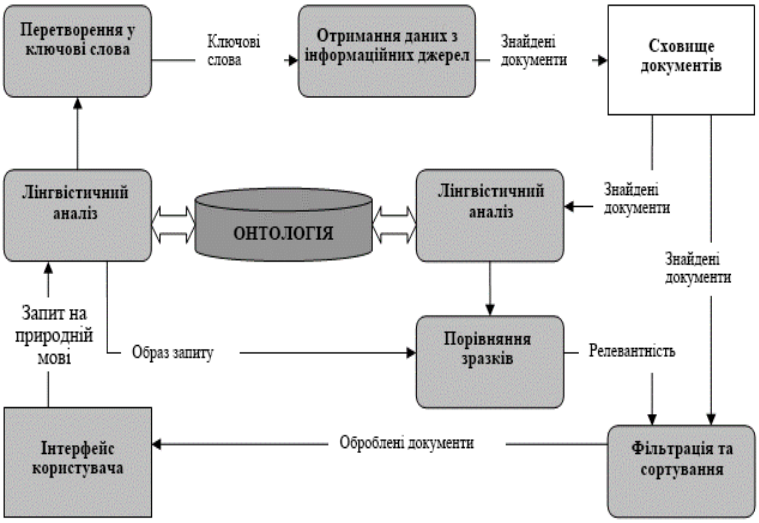
пов'язане з формуванням нормативного профілю, є актуально і передбачає, що пошук буде здійснюватися за допомогою онтологічної моделі тексту.

- Користувач вводить запит, який піддається семантичному аналізу, розширюється за рахунок побудованої граматики, потім перетворюється в ключові слова і передається пошуковій машині.
- Пошукова машина повертає знайдені документи, вони також піддаються лінгвістичному розбору і формуються семантичні образи документів.
- Образи документів порівнюються з образом запиту, робиться висновок про релевантність кожного з документів і результати аналізу (документи, які були визнані релевантними) надаються користувачеві.

6

Діаграма потоків даних онтологічної моделі

Застосування моделей нечіткої атрибутної граматики



7

Завданням дослідження

є автоматизована обробка тексту документа на основі онтологічної моделі.

На даному етапі розвитку методів аналізу тексту не представляється можливим аналізувати природний мову без будь-яких обмежень, тому **необхідно** виявити особливості існуючої практики написання ТД і сформулювати додаткові вимоги.

- В результаті було відзначено, що ТД є документ, написаний технічною мовою.
  ТД відповідає наступним особливостям стилю:
- текст не містить образних виразів, оціночних прикметників, майже позбавлений говірок,
- природна полісемічність мови зводиться до мінімуму використанням заздалегідь визначених термінів;
- текст містить наступні граматичні конструкції: граматична основа з рядом доповнень (домінуюча конструкція), причетні і дієприслівникові обороти.
- ТД на природній мові містить семантично значиму інформацію, яку і слід формалізувати і представити у вигляді інваріантної до ПМ моделі.

8

Семантична модель тексту ТД

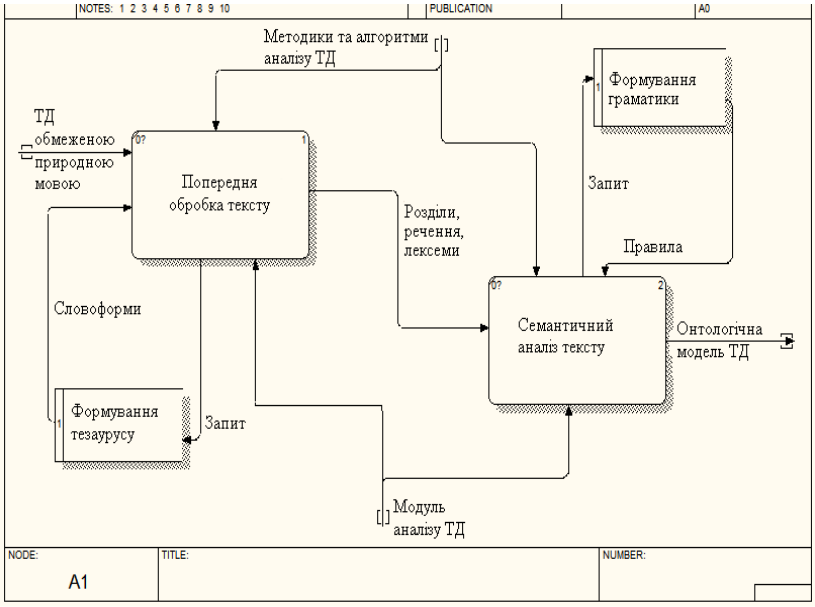
містить розроблену
розширену
нечітку атрибутну
граматику
над онтологічною
структурою
формального
документу

Методика аналізу тексту технічної документації

Розширена нечітка атрибутна граматика, необхідна для автоматизованого аналізу тексту технічного завдання, визначена у вигляді:

$$AG = <N, T, P, S, B, F, A, R (A)>,$$

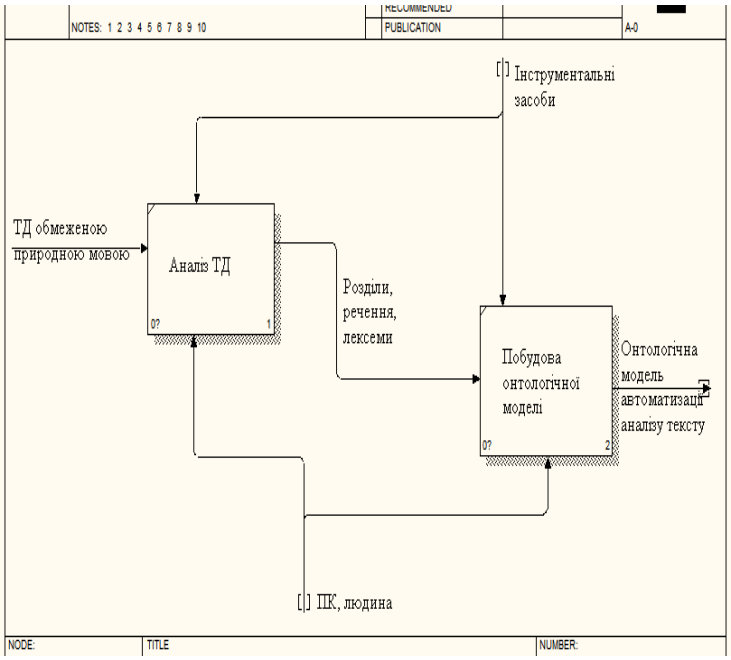
Діаграма аналізу тексту ТД



Функціональні моделі аналізу тексту технічної документації

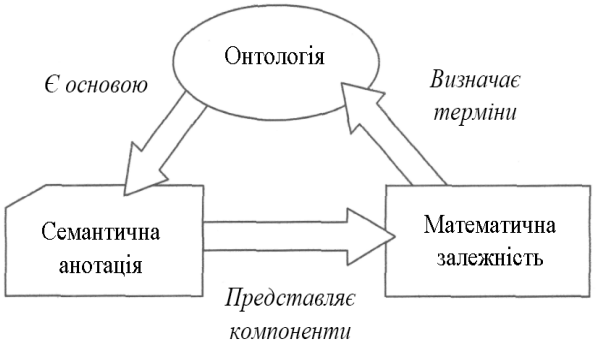
- Для семантичного аналізу необхідна попередня обробка тексту ТД, яка представлена на ПМ, а також використовуються методики та алгоритми лексичного аналізу

  Для побудови онтологічної моделі необхідно вирішити задачу аналізу ТД.
- На вхід подається ТД на ПМ, а керуючими механізмами є методики і алгоритми аналізу ТД, підсистема формалізації ТД, модуль аналізу ТД і алгоритми формалізації ТД.



Діаграма формального аналізу тексту ТД

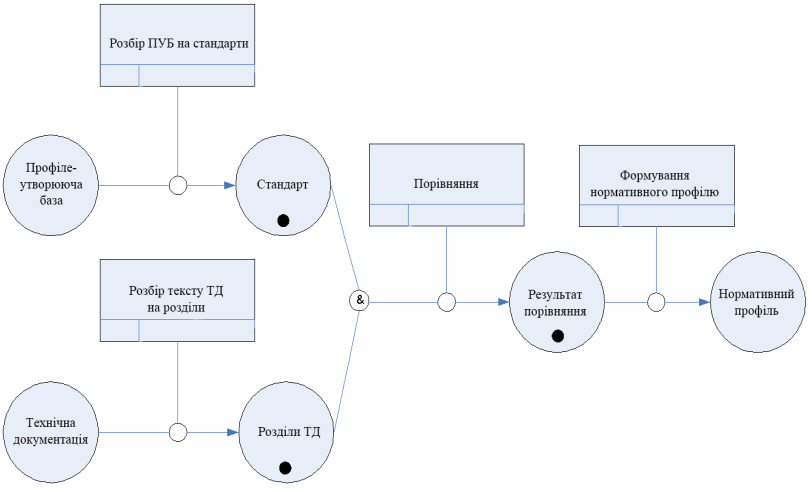
створювана онтологія описує знання про терміни, які використовуються у визначенні математичного виразу, а відповідна семантична анотація вказує на спосіб реалізації математичної залежності, що описана в онтологічній моделі тексту



13

Розробка онтологічної моделі семантичного аналізу тексту

Опис онтологічного інжинірингу при формуванні семантичного опису математичної залежності

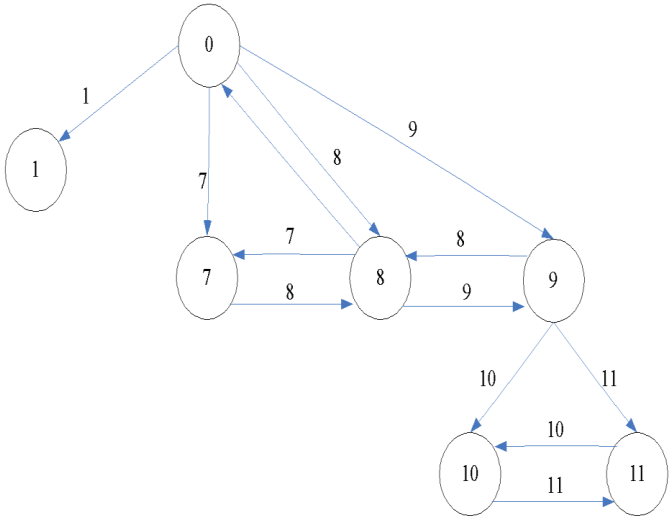


Діаграма стану онтологічної моделі семантичного аналізу тексту технічної документації, стандарт IDEF5

14

Розробка алгоритму автоматичної побудови онтологій

Кінцевий автомат для онтологічної моделі тексту



15

Алгоритмічне забезпечення побудови онтологічної моделі тексту

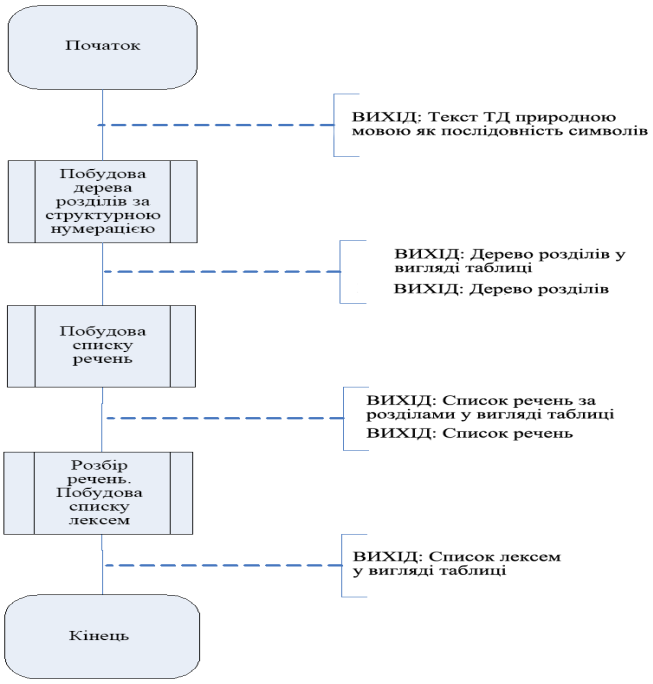
- Аналіз ТД складається з двох блоків: попередня обробка тексту ТД і семантичний аналізатор.

- Блок попередньої обробки тексту з вихідного тексту ТД генерує список лексем обмеженого природної мови технічної документації.
- Семантичний аналізатор приймає список лексем і обробляє його згідно граматиці ТД, генеруючи онтологічну структуру у вигляді бази знань
- Попередня обробка тексту призначена для того, щоб розділити вихідний текст на природній мові на окремі лексеми, ця операція виконується в три етапи: поділ на розділи, пропозиції і окремі лексеми.
- Після першого етапу обробляється не весь текст ТД, а його частини, що представлені по розділах.
- По ходу роботи лексичного аналізатора текст ТД дробиться спочатку на все більш дрібні розділи, потім на окремі пропозиції (зі збереженням структури розділів) і лексеми із зазначенням приналежності до пропозицій.

16

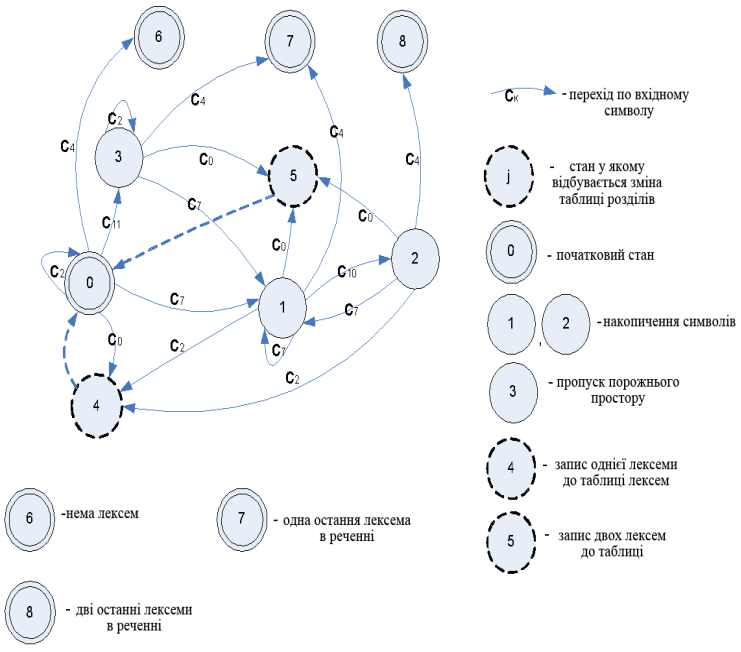
Поділ технічної документації на розділи здійснюється за допомогою кінцевого автомата

Алгоритм попередньої обробки тексту



17

Розроблено кінцевий автомат для розбору на лексеми



18

Пошук

Пошук – операція, що виконує пошук в бібліотеці онтологічного додатку, що містить терміни онтологічного запиту, а також ТД і СТ

$$V = \{\text{Term}\},\ \text{Term} = (\text{ID}, \text{KeywordSet}, \text{Description})$$

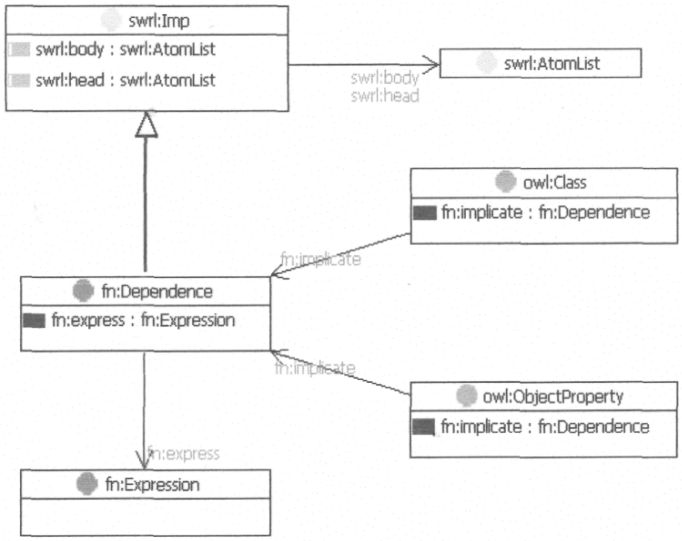
терміни запиту відображаються в словнику бібліотеки, визначеному як множина термів

Реалізація онтологічної моделі тексту

Передбачається, що база знань реалізована засобами стандарту Semantic Web – за допомогою Protege.

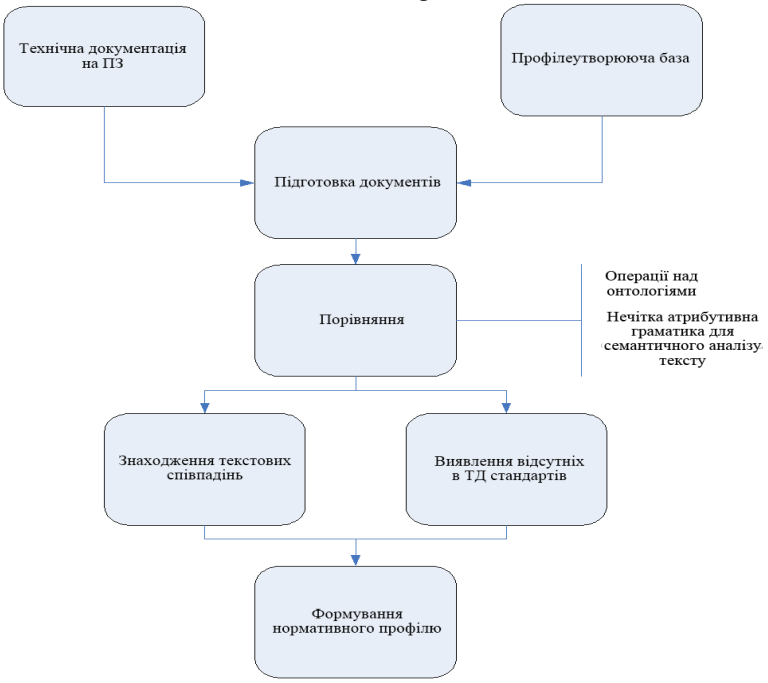
Для реалізації, онтологія тексту для автоматизації аналізу тексту при формуванні нормативного профілю інтегрується в модель *Ф* математичної залежності в онтологічну базу знань, пропонується розширити моделі семантичного опису ресурсів і представлення логічних правил SWRL метамодель на основі синтаксису Protege,

Модель інтеграції семантичної анотації математичної залежності в онтологію в термінах мови OWL



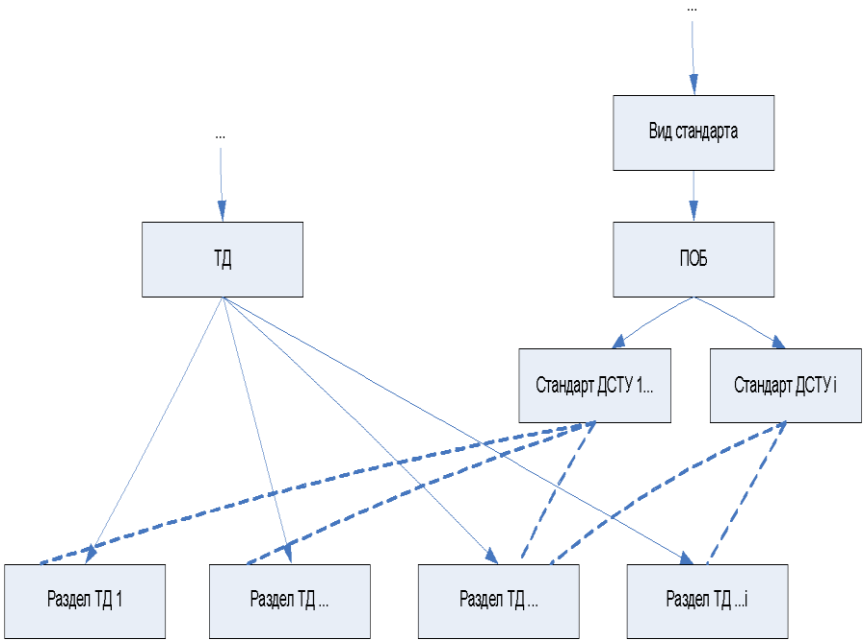
21

Загальна схема онтологічної моделі тексту для автоматизації семантичного аналізу технічної документації



22

Розширення онтологічної моделі



23

Вибір середовища розробки

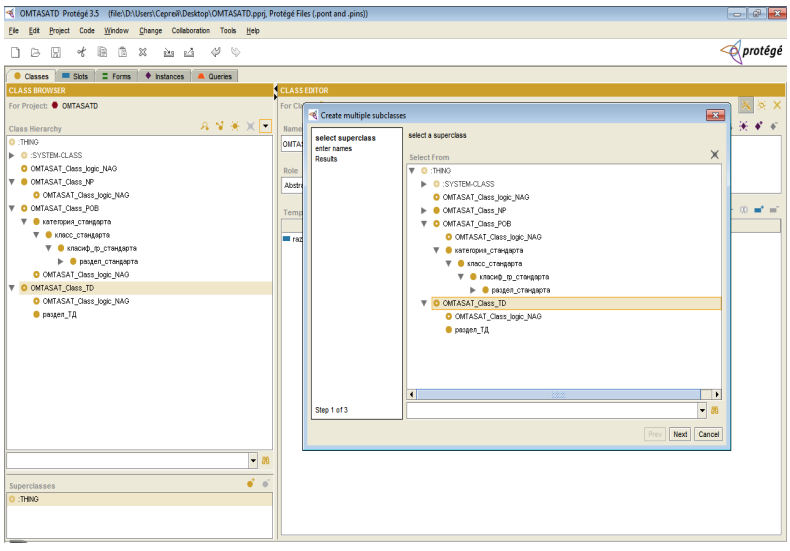
- Protégé – це вільний, відкритий редактор онтологій і фреймворк для побудови баз знань.
- Платформа Protégé підтримує два основних способи моделювання онтологій за допомогою редакторів Protégé-Frames і Protégé-OWL.
- Онтології, побудовані в Protégé, можуть бути експортовані в декілька форматів, включаючи RDF (RDFSchema), OWL і XMLSchema.
- Protégé має відкриту, легко розширювану архітектуру за рахунок підтримки модулів розширення функціональності

Схема базових класів онтології

Для тестового прикладу ТД і СТ були взяті з державних стандартів «Єдина система програмної документації»

25

Приклад завантаження ТД



26

На прикладі онтології проілюстровані можливості обробки онтологічної моделі тексту в середовищі Protégé

- Значно скорочується час сертифікації, що витрачається на формування НП до ТД, за допомогою використання онтологічної моделі тексту.

- Підвищення ефективності полягає в значному скороченні часу аналізу тексту ТД (від 36 до 57% залежно від обсягу ТД).

ВИСНОВКИ

Використання онтологічних моделей та структурованість бази знань, насиченість її термінами і дефініціями дозволяють використовувати найбільш продуктивні – текстологічні методи видобування знань, які створюють передумови побудови автоматизованих процедур обробки текстів.

Висока трудомісткість побудови, необхідність наявності певних навичок створення і масовість онтологій для представлення знань в процесі сертифікації вимагають розробки спеціальних методів і алгоритмів обробки текстових документів.

- Проведено аналіз існуючих проблем, пов'язаних із застосуванням онтологічних моделей тексту при автоматизації процесу семантичного аналізу технічної документації
- Висунуто вимоги до онтологічної моделі тексту для автоматизації семантичного аналізу тексту технічної документації: до її структури, наповнення і функціонального призначення.
- Представлено автомат розбору тексту ТД на розділи.
- Спроектована автоматизована система семантичного аналізу тексту ТД, розроблена загальна архітектура, функціональна структура автоматизованої системи, представлена структура вихідних файлів, показані діаграми класів.
- Розглянуто середовище моделювання онтологічних структур – Protégé, як засіб розробки, що дозволяє розширювати себе для вирішення більш широкого кола завдань.
- Проілюстровані можливості побудови онтологічної моделі тексту в середовищі Protégé.

ДОДАТОК В

Апробація роботи


CERTIFICATE
is awarded to
Vavilov Pavlo
for being an active participant in
IX International Scientific and Practical Conference
"TOPICAL ISSUES OF THE DEVELOPMENT OF MODERN SCIENCE"
24 Hours of Participation
SOFIA
6-8 May 2020
sci-conf.com.ua
INTERNATIONAL SCIENTIFIC AND PRACTICAL CONFERENCE
TOPICAL ISSUES OF THE DEVELOPMENT OF MODERN SCIENCE
SCI-CONF.COM.UA
SOFIA
6-8 MAY 2020