МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені І.І.МЕЧНИКОВА

(повне найменування вищого навчального закладу)

Інститут математики, економіки та механіки

(повне найменування інституту, назва факультету (відділення))

Кафедра математичного забезпечення комп'ютерних систем

(повна назва кафедри (предметної, циклової комісії))

Дипломна робота

бакалавра

(освітньо-кваліфікаційний рівень)

на тему Система метамоделирования предметной области.
Подсистема построения модели ПрО

Система метамоделювання предметної області.
Підсистема побудови моделі ПрО

Subject domain metamodeling system.
SD model analysis subsystem

Виконав: студент 4 курсу, групи 1
напряму підготовки 6.050102 –
«Комп'ютерна інженерія»

(шифр і назва напряму підготовки, спеціальності)

Куніцин А.С.

(прізвище та ініціали)

Керівник
д.т.н., проф. Малахов Є.В.

(прізвище та ініціали)

Рецензент
к.ф.-м.н., доц. Шпинарьова І.М.

(прізвище та ініціали)

Рекомендовано до захисту:

Протокол засідання кафедри

№_____ від «___»_____р.

Завідувач кафедри

_____ _____

(підпис) (прізвище, ініціали)

Захищено на засіданні ЕК № ___

протокол №___ від «___» _____р.

Оцінка _____

(за 4-х бальною шкалою, за шкалою ECTS, бал.)

Голова ЕК

_____ О.О. Арсірій

(підпис) (прізвище, ініціали)

Одеса – 2017

АННОТАЦИЯ

При современном развитии информационных технологий одной из задач, решение которой позволяет повысить эффективность управления любой предметной областью, является задача анализа этой предметной области.

В литературе описаны различные алгоритмы анализа предметных областей. Соответственно, целью данной дипломной работы является разработка приложения, ориентированного на анализ предметной области путём реализации алгоритма поиска определяющих атрибутов и алгоритма классификации связей. Данный анализ обеспечивает выявление в предметной области элементов и характеристик, которые помогут решать возникающие массовые проблемы и в перспективе управлять ими.

Для достижения поставленной цели спроектирована и создана БД, где каждая таблица соответствует определенному элементу предметной области. Также, разработано web-приложение, обеспечивающее взаимодействие пользователя с этой БД. В рамках приложения создан удобный интерфейс, который позволяет визуализировать результаты анализа предметной области.

Кроме того, в процессе работы проведена модификация алгоритма поиска определяющих атрибутов предметной области, через которые осуществляется воздействие на данную предметную область.

На данный момент, результат анализа, получаемый разработанной системой, может быть применен для определения состояния предметной области, её прогнозирования, а в перспективе, благодаря развитию данной системы появится возможность управлять соответствующей предметной областью.

Результаты работы докладывались на 14 Всеукраинской конференции студентов и молодых учёных «Информатика, информационные системы и технологии».

АНОТАЦІЯ

При сучасному розвитку інформаційних технологій одним із завдань, вирішення якого дозволяє підвищити ефективність управління будь-якою предметною областю, є завдання аналізу цієї предметної області.

В літературі описані різні алгоритми аналізу предметних областей. Відповідно, метою даної дипломної роботи є розробка програми, орієнтованого на аналіз предметної області шляхом реалізації алгоритму пошуку визначальних атрибутів і алгоритму класифікації зв'язків. Даний аналіз забезпечує виявлення в предметної області елементів і характеристик, які допоможуть вирішувати виникаючі масові проблеми і в перспективі управляти ними.

Для досягнення поставленої мети спроектована і створена БД, де кожна таблиця відповідає певному елементу предметної області. Також, розроблен web-додаток, що забезпечує взаємодію користувача з цієї БД. В рамках програми створено зручний інтерфейс, який дозволяє візуалізувати результати аналізу предметної області.

Окрім того, в процесі роботи проведена модифікація алгоритму пошуку визначальних атрибутів предметної області, через які здійснюється вплив на дану предметну область.

На даний момент, результат аналізу, одержуваний розробленою системою, може бути застосований для визначення стану предметної області, її прогнозування, а в перспективі, завдяки розвитку даної системи з'явиться можливість керувати відповідною предметною областю.

Результати роботи доповідалися на 14 Всеукраїнської конференції студентів та молодих вчених «Інформатика, інформаційні системи і технології».

ABSTRACT

According to the IT modern development one of the tasks whose solution gives an opportunity to increase the any subject domain management efficiency is the problem of this subject domain analyze.

Nowadays, there are various algorithms of the subject domains analisys. The purpose of this graduate work is the development of an application whish is focused on the analysis of the subject domain by implementing the algorithm of extracting of the essential attributes and the algorithm of connection classification. This analysis provides the extracting of the elements and the characteristics which helps to solve mass problems and to control them.

To achieve this purpose, a database was designed and created. Each table of this database corresponds to a specific element of the subject domain. Also, it was developed a web-application which provides user interaction with this database. It was created a user-friendly interface which allows to visualize the results of the subject domain analysis.

In development progress also the algorithm of extracting of the essential attributes was modificated.

The obtained results can be applied to determine the state of the subject domain, its prediction. In future it will be possible to manage the relevant subject domain.

The results of the work were reported at the 14th All-Ukrainian Conference of Students and Young Scientists "Informatics, Information Systems and Technologies".

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ І ТЕРМІНІВ

ПрО – предметна область;

ПС – продукційна система;

ACID – атомарність, узгодженість, ізольованість і надійність;

REST – Representational State Transfer;

MVC – Model View Controller;

JDBC – Java Database Connectivity;

ЗМІСТ

стр.

ВСТУП

Інформаційні системи широко використовуються в більшості предметних областей (ПрО). Вони значно полегшують людську працю, автоматизуючи виконання різних функцій або заміщаючи людину зовсім.

Дотримуючись одного з визначень, ПрО можна розглядати як частину реального світу, що описується згідно з установленими критеріями [1], або це можуть бути знання, що використовуються для позначення області будь-якої людської діяльності. У ПрО існують різні проблеми, які можуть бути сформульовані, проаналізовані та вирішені в міру можливості. Для вирішення цих проблем, а також розв’язання задачі управління, необхідно виконувати аналіз ПрО і її станів. У свою чергу це вимагає програмної реалізації інформаційно-структурної моделі ПрО і математичних операцій над нею.

Дослідження і аналіз ПрО значно полегшать управління і роботу з нею [2]: дозволять виявляти елементи, існування або відсутність яких в певній мірі впливає на функціонування ПрО, що дасть можливість підвищити ефективність роботи в певній сфері людської діяльності. Також, результатом аналізу може стати прогнозування: на підставі наявності достатньої кількості даних можна робити висновки щодо майбутнього деяких атрибутів ПрО.

З точки зору реального світу, метою аналізу є виявлення, класифікація і формалізація інформації про всі аспекти ПрО, яка потрібна для роботи вузького кола фахівців. Такими можуть бути системні аналітики або бізнес-аналітики, які в процесі аналізу отримують великі обсяги інформації, з яких треба відібрати істотну частину, а також вміти знаходити в ній проблеми — області, де діяльність не відповідає поставленим завданням. Автоматизація аналізу повинна забезпечити відбір цієї суттєвої частини і пошуку рішень масових проблем.

Метою даної роботи є розробка програми, заснованої на такій моделі ПрО, яка дозволить автоматизувати класифікацію та аналіз ПрО шляхом реалізації відповідних алгоритмів.

Для досягнення поставленої мети необхідно розв’язати наступні задачі:

1) виконати огляд і модифікацію вже існуючих алгоритмів аналізу ПрО;

2) розробити модель даних у вигляді інформаційної моделі для представлення ПрО;

3) розробити web-додаток, який дозволить проводити аналіз стану ПрО.

ВИСНОВОК

У даній роботі виконана реалізація алгоритмів аналізу ПрО. Для розв’язання поставлених завдань обраний спосіб представлення ПрО, запропонований в публікаціях [1], де ПрО представлена у вигляді кортежу, що складається з трьох основних елементів: об'єктів, зв'язків і масових проблем. Дана концепція запропонована для реалізації за допомогою РБД, що забезпечує зручне зберігання і цілісність даних.

На підставі запропонованої інформаційної моделі спроектована БД, яка в повній мірі відповідає математичній моделі ПрО. Для реалізації були обрані два алгоритми: алгоритм пошуку визначальних атрибутів і алгоритм класифікації зв'язків, які дозволяють виявити в ПрО атрибути, зміна або видалення яких може значно вплинути на стан ПрО та дають зрозуміти, за яких умов дані зв'язки існують, а за яких відсутні.

Отже, розроблено web-додаток, який реалізує модель ПрО, яка зберігається в спроєктованій БД. Для реалізації програми була обрана мова програмування Java і ряд технологій, які дозволили створити повноцінний RESTful web-сервіс. Це забезпечило додатку широкий спектр для подальшої розробки, будь то додавання нових модулів або внесення змін до логіки існуючих.

Планується подальшу роботу вести в наступних напрямках: для підвищення продуктивності алгоритмів – їх розпаралелювання, а для підвищення точності аналізу – реалізацію алгоритму аналіза масових проблем.

СПИСОК ВИКОРИСТАННИХ ДЖЕРЕЛ

1. Малахов Е.В., Расширение операций над метамоделями предметных областей с учётом массовых проблем // Восточно-европейский журнал передовых технологий. – Харьков, 2010. – Вып. 5/2 (47). – С.20-24.
2. Куницын А.С., Построение модели предметной области. Анализ предметной области/ А.С. Куницын, Е.В. Малахов, Д.О. Щелконогов // Тези докладів чотирнадцятої всеукраїнської конференції студентів и молодих науковців «Інформатика, інформаційні системи і технології». Одеса, 14 квітня 2017 р. – Одеса, 2017. – С.162-165.
3. МАССОВАЯ ПРОБЛЕМА [Електронний ресурс] – Режим доступа: http://dic.academic.ru/dic.nsf/enc_mathematics/3025/МАССОВАЯ.– 15.05.17
4. Mezhuyev Vitaliy, The Method and Algorithms to Find Essential Attributes and Objects of Subject Domains / Vitaliy Mezhuyev, Eugene Malakhov, Denys Shchelkonogov // IEEE 2015 International Conference on Computer, Communication, and Control Technology (I4CT 2015), April 21-23, 2015. – Kuching, Sarawak, Malaysia, 2015. – PP. 310-314.
5. Malakhov E., Stability and Inevitability of Connections of Subject Domains / E. Malakhov, D. Shchelkonogov // Электротехнические и компьютерные системы. – О.: „Наука и Техника“, 2015. – № 19 (95), 2015. – С. 174 – 177.
6. Three-tier architectures [Електронний ресурс] – Режим доступа: https://www.ibm.com/support/knowledgecenter/en/SSAW57_8.5.5/com.ibm.websphere.nd.doc/ae/covr_3-tier.html. – 16.05.17
7. Малахов Е.В. Организация баз данных: конспект. – О.: ОНУ, 2014.
8. Spring Framework [Електронний ресурс] – Режим доступа: http://docs.spring.io/spring/docs/current/spring-framework-reference/html/. – 17.05.17
9. Apache Maven [Електронний ресурс] – Режим доступа: http://www.apache-maven.ru/. – 17.05.17

10. Архитектура REST [Електронний ресурс] – Режим доступа: https://habrahabr.ru/post/38730/. – 17.05.17
11. PostgreSQL [Електронний ресурс] – Режим доступа: https://www.postgresql.org/about/. – 17.05.17
12. JDBC [Електронний ресурс] – Режим доступа: http://java.cnam.fr/iagl/biblio/spec/jdbc-3_0-fr-spec.pdf. – 17.05.17
13. JSQLParser [Електронний ресурс] – Режим доступа: https://github.com/JSQLParser/JSqlParser. – 19.05.17

ДОДАТОК А

СКРИПТЫ СТВОРЕННЯ ТАБЛИЦЬ

```
// оголошення перевантажаємої змінної
DECLARE @T;

// скрипт створення таблиці Subject_domain
SET @T = 'Subject_domain';
CREATE TABLE @T (sd_id integer not null primary key,
sd_name text not null);

// скрипт створення таблиці SD_state
SET @T = 'SD_state';
CREATE TABLE @T (sd_state_id integer not null primary key,
sd_id integer not null references Subject_domain(sd_id),
timestamp integer not null);

// скрипт створення таблиці Object
SET @T = 'Object';
CREATE TABLE @T (object_id integer not null primary key,
sd_id integer not null references Subject_domain(sd_id),
obj_name text not null);

// скрипт створення таблиці Object_instance
SET @T = 'Object_instance';
CREATE TABLE @T (object_instance_id integer not null primary
key,
object_id integer not null references Object(object_id),
object_instance_name text not null);

// скрипт створення таблиці Connection
SET @T = 'Connection';
CREATE TABLE @T (connection_id integer not null primary key,
sd_id integer not null references Subject_domain(sd_id),
connection_name text not null);

// скрипт створення таблиці Attribute
SET @T = 'Attribute';
CREATE TABLE @T (attribute_id integer not null primary key,
object_id integer not null references Object(object_id),
attribute_name text not null);

// скрипт створення таблиці Attribute_value
SET @T = 'Attribute_value';
```

```
CREATE TABLE @T (attribute_value_id integer not null primary
key,
attribute_id integer not null references
Attribute(attribute_id),
attribute_value integer not null);

// скрипт створення таблиці Object_state
SET @T = 'Object_state';
CREATE TABLE @T (object_instance_id integer not null references
Object_instance(object_instance_id),
attribute_value_id integer not null references
Attribute_value(attribute_value_id),
sd_state_id integer not null references SD_state(sd_state_id));

// скрипт створення таблиці Connection_instance
SET @T = 'Connection_instance';
CREATE TABLE @T (connection_instance_id integer not null primary
key,
connection_id integer not null references
Connection(connection_id),
connection_instance_name text not null);

// скрипт створення таблиці Object_instance_list
SET @T = 'Object_instance_list';
CREATE TABLE @T (connection_instance_id integer not null
references Connection_instance(connection_instance_id),
object_instance_id integer not null references
Object_instance(object_instance_id));

// скрипт створення таблиці Object_list
SET @T = 'Object_list';
CREATE TABLE @T (connection_id integer not null references
Connection(connection_id),
object_id integer not null references Object(object_id));

// скрипт створення таблиці Connection_state
SET @T = 'Connection_state';
CREATE TABLE @T (connection_state_id integer not null primary
key,
sd_state_id integer not null references SD_state(sd_state_id),
cconnection_instance_id integer not null references
Connection_instance(connection_instance_id),
connection_state_name text not null);

// скрипт створення таблиці Mass_problem
SET @T = 'Mass_problem';
```

```
CREATE TABLE @T (mass_problem_id integer not null primary key,
sd_id integer not null references Subject_domain(sd_id),
mass_problem_name text not null);

// скрипт створення таблиці Parameter
SET @T = 'Parameter';
CREATE TABLE @T (parameter_id integer not null primary key,
mass_problem_id integer not null references
Mass_problem(mass_problem_id),
parameter_name text not null);

// скрипт створення таблиці Individual_task
SET @T = 'Individual_task';
CREATE TABLE @T (individual_task_id integer not null primary
key,
mass_problem_id integer not null references
Mass_problem(mass_problem_id),
priority integer not null
individual_task text not null);

// скрипт створення таблиці Parameter_value
SET @T = 'Parameter_value';
CREATE TABLE @T (parameter_value_id integer not null primary
key,
individual_task_id integer not null references
Individual_task(individual_task_id),
parameter_id integer not null references
Parameter(parameter_id),
parameter_value integer not null);
```

ДОДАТОК Б

КОД АЛГОРИТМУ ПОШУКУ ВИЗНАЧАЛЬНИХ АТРИБУТІВ

```
package diploma.logic.algos;

import diploma.logic.graphs.Graph;
import diploma.logic.graphs.Vertex;
import diploma.logic.graphs.VertexConnection;
import diploma.logic.graphs.prodsys.entities.DefiningAttribute;
import diploma.logic.parsers.entities.QueryAttribute;

import java.math.BigDecimal;
import java.math.RoundingMode;
import java.util.ArrayList;
import java.util.Collection;
import java.util.Map;
import java.util.Queue;
import java.util.concurrent.ConcurrentHashMap;
import java.util.concurrent.ConcurrentLinkedQueue;
import java.util.concurrent.ConcurrentSkipListSet;
import java.util.concurrent.atomic.AtomicInteger;

public class AcyclicDownTopAlgorithm {

    private Collection<DefiningAttribute> definingAttributes;
    private Map<Vertex<QueryAttribute>, Double>
attributesMeasures;
    private Map<Vertex<QueryAttribute>, AtomicInteger>
numberOfUnprocessedChildren;
    private Queue<Vertex<QueryAttribute>> vertexesToProcess;
    private AtomicInteger numberOfUnprocessedVertexes;

    public AcyclicDownTopAlgorithm() {
        definingAttributes = new
ConcurrentSkipListSet<DefiningAttribute>();
        numberOfUnprocessedChildren = new
ConcurrentHashMap<Vertex<QueryAttribute>, AtomicInteger>();
        vertexesToProcess = new
ConcurrentLinkedQueue<Vertex<QueryAttribute>>();
        attributesMeasures = new
ConcurrentHashMap<Vertex<QueryAttribute>, Double>();
    }

// метод, що ініціалізує початок роботи алгоритму
```

```
    public Collection<DefiningAttribute>
getDefiningAttributes(Graph graph) {
        try {
            Graph<QueryAttribute> attributesGraph = graph;
            initializeFields(attributesGraph);
            calculateMeasures();
            fillDefiningAttributesCollection();
        } catch (Exception e) {
            e.printStackTrace();
        }

        return definingAttributes;
    }

    private void fillDefiningAttributesCollection() {
        definingAttributes = new ArrayList<DefiningAttribute>();
        for (Map.Entry<Vertex<QueryAttribute>, Double>
vertexDoubleEntry : attributesMeasures.entrySet()) {
            QueryAttribute attribute =
vertexDoubleEntry.getKey().getValue();
            double measure = vertexDoubleEntry.getValue();
            DefiningAttribute definingAttribute = new
DefiningAttribute(attribute, measure);
            definingAttributes.add(definingAttribute);
        }
    }

// метод, що ініціалізує вершини графу

    private void initializeFields(Graph<QueryAttribute>
attributesGraph) {

        Collection<Vertex<QueryAttribute>> vertexes =
attributesGraph.getVertexes();

        numberOfUnprocessedVertexes = new
AtomicInteger(vertexes.size());

        for(Vertex<QueryAttribute> vertex : vertexes){
            Integer numberOfChildren =
calculateNumberOfChildren(vertex);
            numberOfUnprocessedChildren.put(vertex, new
AtomicInteger(numberOfChildren));
            if(numberOfChildren == 0){
                vertexesToProcess.add(vertex);
            }
```

```
        }
    }

    private Integer
calculateNumberOfChildren(Vertex<QueryAttribute> vertex) {
        return vertex.getOutgoingConnections().size();
    }

// метод підліку всіх числових значень

    private void calculateMeasures() {

        while(notAllVertexesWereProcessed()){
            if(vertexesToProcess.size() > 0){
                Vertex<QueryAttribute> nextVertex =
getVertexAndUpdateCounter();
                calculateMeasure(nextVertex);
                updateParents(nextVertex);
            }
        }

    }

    private Vertex<QueryAttribute> getVertexAndUpdateCounter() {
        Vertex<QueryAttribute> nextVertex =
vertexesToProcess.remove();
        numberOfUnprocessedVertexes.decrementAndGet();
        return nextVertex;
    }

    private void updateParents(Vertex<QueryAttribute>
nextVertex) {
        Collection<VertexConnection<QueryAttribute>> ingoing =
nextVertex.getIngoingConnections();
        for (VertexConnection<QueryAttribute> connection :
ingoing) {
            Vertex<QueryAttribute> parent =
connection.getParent();
            AtomicInteger numberOfRemainingChildren =
numberOfUnprocessedChildren.get(parent);
            int updatedValue =
numberOfRemainingChildren.decrementAndGet();
            if(updatedValue == 0){
                vertexesToProcess.add(parent);
            }
        }
```

```
    }

// метод підліку числового значення певної вершини

    private void calculateMeasure(Vertex<QueryAttribute>
nextVertex) {
        double measure = 1;
        Collection<VertexConnection<QueryAttribute>> connections
= nextVertex.getOutgoingConnections();
        for (VertexConnection<QueryAttribute> connection :
connections) {
            double childMeasure =
getPartOfChildMeasure(connection);
            measure += childMeasure;
        }
        attributesMeasures.put(nextVertex, measure);
    }

// метод підсомування числових значень, враховючи числові
значення вершин-потомків

    private double
getPartOfChildMeasure(VertexConnection<QueryAttribute>
connection) {
        Vertex<QueryAttribute> vertex = connection.getChild();
        double measure = attributesMeasures.get(vertex);
        int numberOfParentsOfChild =
vertex.getIngoingConnections().size();
        double result = measure / numberOfParentsOfChild;
        result = new BigDecimal(result).setScale(1,
RoundingMode.UP).doubleValue();
        return result;
    }

    private boolean notAllVertexesWereProcessed() {
        return numberOfUnprocessedVertexes.get() != 0;
    }
}
```

ДОДАТОК В

КОД АЛГОРИТМУ КЛАСИФІКАЦІЇ ЗВ’ЯЗКІВ

```
package diploma.logic.algos;

import diploma.logic.algos.entities.*;

import java.util.*;

public class ConnectionAlgorithm {

    private Set<AlgoConnection> connectionSet;
    private List<List<AlgoConnectionInstance>>
connectionInstanceLists;
    private List<AlgoMassProblem> massProblemList;
    private Map<AlgoConnection, Set<AlgoConnectionInstance>>
groupedConnectionInstances;
    private Map<AlgoConnection, Double> stability,
inevitability;
    private Map<AlgoMassProblem,
List<List<AlgoConnectionInstance>>> groupedMassProblems;
    private Map<AlgoMassProblem, Map<AlgoConnection,
Set<AlgoConnectionInstance>>> groupedMassProblemsConnections;
    private Map<AlgoMassProblem, Map<AlgoConnection, Double>>
createProbability, destroyProbability;

    public ConnectionAlgorithm(){
        connectionSet = new LinkedHashSet<AlgoConnection>();
        connectionInstanceLists = new
ArrayList<List<AlgoConnectionInstance>>();
        groupedConnectionInstances = new
LinkedHashMap<AlgoConnection, Set<AlgoConnectionInstance>>();
        stability = new LinkedHashMap<AlgoConnection, Double>();
        inevitability = new LinkedHashMap<AlgoConnection,
Double>();
        massProblemList = new ArrayList<AlgoMassProblem>();
        groupedMassProblems = new LinkedHashMap<AlgoMassProblem,
List<List<AlgoConnectionInstance>>>();
        groupedMassProblemsConnections = new
LinkedHashMap<AlgoMassProblem, Map<AlgoConnection,
Set<AlgoConnectionInstance>>>();
        createProbability = new LinkedHashMap<AlgoMassProblem,
Map<AlgoConnection, Double>>();
        destroyProbability = new LinkedHashMap<AlgoMassProblem,
Map<AlgoConnection, Double>>();
    }

// метод, що ініціалізує початок роботи алгоритму

    public void calculate(Set<AlgoConnection> connectionSet,
                          List<List<AlgoConnectionInstance>>
connectionInstanceLists,
```

```
                            List<AlgoMassProblem>
massProblemList){
        this.connectionSet = connectionSet;
        this.connectionInstanceLists = connectionInstanceLists;
        this.massProblemList = massProblemList;

        groupByConnection();
        calculateStability();
        calculateInevitability();
        calculateMassProblemsScales();
        showStat();
    }

// метод групування екземплярів по зв’язкам

    private void groupByConnection(){
        for(AlgoConnection connection : connectionSet){
            groupedConnectionInstances.put(connection,
findByConnection(connection));
        }
    }

//метод пошуку екземплярів по їх зв’язкам

    private Set<AlgoConnectionInstance>
findByConnection(AlgoConnection connection){
        Set<AlgoConnectionInstance> groupedConnectionInstanceSet
= new LinkedHashSet<AlgoConnectionInstance>();

        for(List<AlgoConnectionInstance> list :
connectionInstanceLists) {
            for (AlgoConnectionInstance connectionInstance :
list) {
                if
(connection.getObj1().getObjectInstanceList().contains(new
AlgoObjInstance(connectionInstance.getField1())) &&

connection.getObj2().getObjectInstanceList().contains(new
AlgoObjInstance(connectionInstance.getField2()))) {

groupedConnectionInstanceSet.add(connectionInstance);
                }
            }
        }

        return groupedConnectionInstanceSet;
    }

// метод підліку стабільності

    private void calculateStability(){
        Set<AlgoConnectionInstance> connectionInstanceSet;
```

```
        for(Map.Entry entry :
groupedConnectionInstances.entrySet()){
            int n = 0, m = 0;
            connectionInstanceSet = (Set)entry.getValue();

            for(AlgoConnectionInstance connectionInstance :
connectionInstanceSet){
                n +=
findRepeatsByConnectionInstance(connectionInstance);
                m +=
findLinkedRepeatsByConnectionInstance(connectionInstance);
            }

            stability.put((AlgoConnection)entry.getKey(),
(double)m/n);
        }
    }

// метод підліку неминучості

    private void calculateInevitability(){
        Set<AlgoConnectionInstance> connectionInstanceSet;
        for(Map.Entry entry :
groupedConnectionInstances.entrySet()){
            int n = 0, m = 0;
            connectionInstanceSet = (Set)entry.getValue();

            for(AlgoConnectionInstance connectionInstance :
connectionInstanceSet){
                n +=
findRepeatsByConnectionInstance(connectionInstance);
                m +=
findEmergenceByConnectionInstance(connectionInstance);
            }

            inevitability.put((AlgoConnection)entry.getKey(),
(double)m/n);
        }
    }

    private int
findRepeatsByConnectionInstance(AlgoConnectionInstance
connectionInstance){
        int n = 0;

        for(List<AlgoConnectionInstance> list :
connectionInstanceLists){
            if(list.contains(connectionInstance)){
                n++;
            }
        }
```

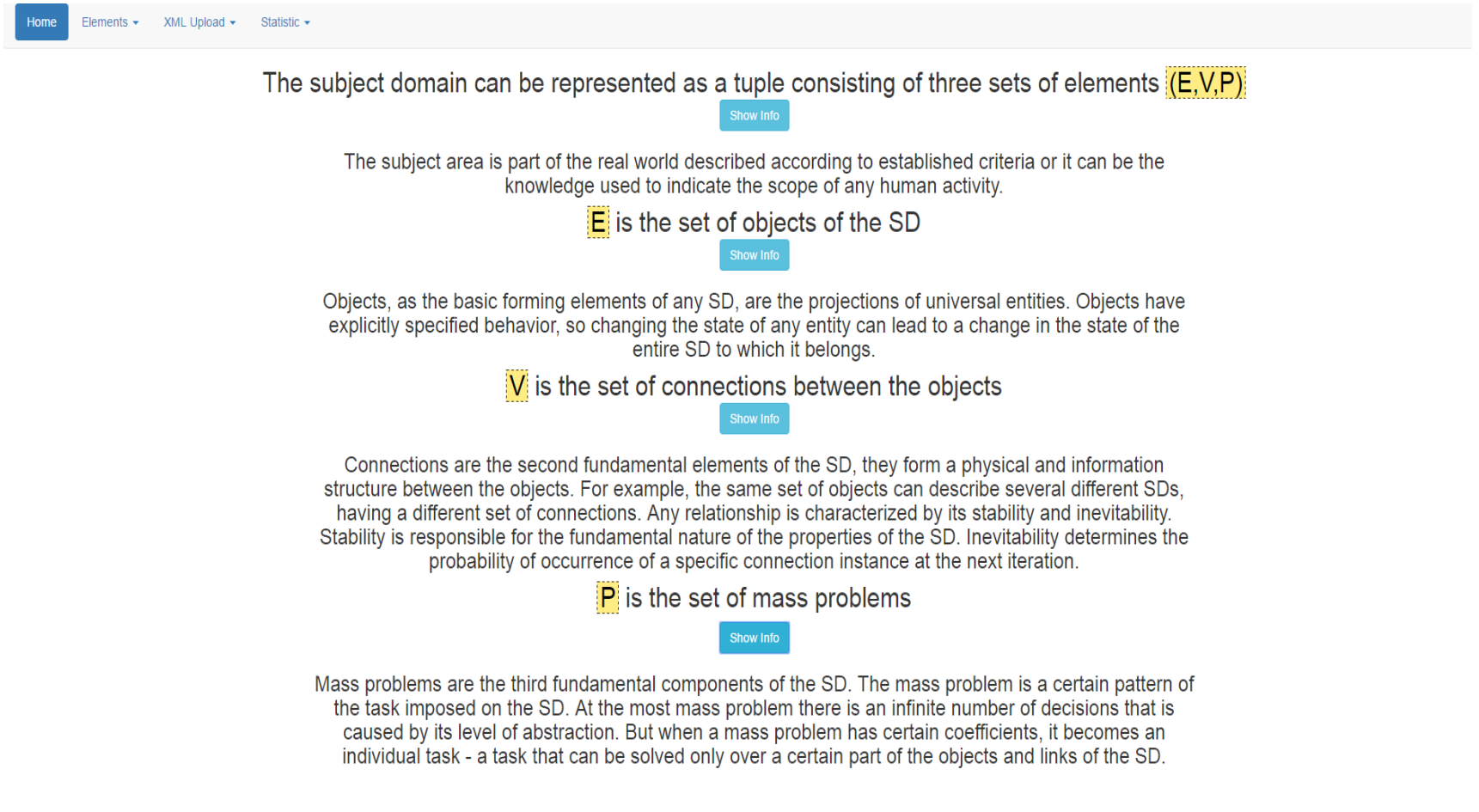
```
        return n;
    }

    private int
findLinkedRepeatsByConnectionInstance(AlgoConnectionInstance
connectionInstance){
        int m = 0;
        boolean repeatFlag = false;

        for(List<AlgoConnectionInstance> list :
connectionInstanceLists){
            if(list.contains(connectionInstance)){
                if(repeatFlag){
                    m++;
                }
                repeatFlag = true;
            } else repeatFlag = false;
        }

        return m;
    }

    private int
findEmergenceByConnectionInstance(AlgoConnectionInstance
connectionInstance){
        int m = 0;
        boolean repeatFlag = false;

        for(List<AlgoConnectionInstance> list :
connectionInstanceLists){
            if(list.contains(connectionInstance)){
                if(!repeatFlag){
                    m++;
                }
                repeatFlag = true;
            } else repeatFlag = false;
        }

        return m;
    }

// метод, що ініціалізує підлік ймовірність появи і пропажі
зв’язку

    public void calculateMassProblemsScales(){
        groupConnectionInstancesByMassProblems();
        groupConnectionsByMassProblems();
        calculateCreateDestroyProbability();
    }

    private void groupConnectionInstancesByMassProblems(){
        for(AlgoMassProblem massProblem : massProblemList){
```

```
            groupedMassProblems.put(massProblem,
getConnectionInstancePairList(massProblem));
        }
    }

    private void groupConnectionsByMassProblems(){
        for(AlgoMassProblem massProblem : massProblemList){
            groupedMassProblemsConnections.put(massProblem,
groupMassProblemConnectionInstancesByConnection(massProblem));
        }
    }

    private List<List<AlgoConnectionInstance>>
getConnectionInstancePairList(AlgoMassProblem massProblem){
        List<List<AlgoConnectionInstance>>
massProblemConnectionInstanceList = new
ArrayList<List<AlgoConnectionInstance>>();

        for(int i = 0; i < massProblemList.size(); i++){
            if(massProblemList.get(i).equals(massProblem)){

massProblemConnectionInstanceList.add(connectionInstanceLists.ge
t(i));

massProblemConnectionInstanceList.add(connectionInstanceLists.ge
t(i + 1));
            }
        }

        return massProblemConnectionInstanceList;
    }

    private Map<AlgoConnection, Set<AlgoConnectionInstance>>
groupMassProblemConnectionInstancesByConnection(AlgoMassProblem
massProblem){
        Map<AlgoConnection, Set<AlgoConnectionInstance>>
groupedMassProblemConnectionInstances = new
LinkedHashMap<AlgoConnection, Set<AlgoConnectionInstance>>();

        for(Map.Entry entry : groupedMassProblems.entrySet()){
            if(entry.getKey().equals(massProblem)){
                for(AlgoConnection connection : connectionSet){

groupedMassProblemConnectionInstances.put(connection,
findByConnection(connection,
(List<List<AlgoConnectionInstance>>)entry.getValue()));
                }
            }
        }

        return groupedMassProblemConnectionInstances;
    }
```

```
    private Set<AlgoConnectionInstance>
findByConnection(AlgoConnection connection,
List<List<AlgoConnectionInstance>>
massProblemConnectionInstancesList){
        Set<AlgoConnectionInstance> connectionInstances = new
LinkedHashSet<AlgoConnectionInstance>();

        for(List<AlgoConnectionInstance> list :
massProblemConnectionInstancesList){
            for(AlgoConnectionInstance connectionInstance :
list){
                if
(connection.getObj1().getObjectInstanceList().contains(new
AlgoObjInstance(connectionInstance.getField1())) &&

connection.getObj2().getObjectInstanceList().contains(new
AlgoObjInstance(connectionInstance.getField2()))) {
                    connectionInstances.add(connectionInstance);
                }
            }
        }

        return connectionInstances;
    }

    private void calculateCreateDestroyProbability(){
        Map<AlgoConnection, Set<AlgoConnectionInstance>>
groupedConnectionInstanceMap;

        for(Map.Entry entry :
groupedMassProblemsConnections.entrySet()){
            groupedConnectionInstanceMap = (Map<AlgoConnection,
Set<AlgoConnectionInstance>>)entry.getValue();
            List<List<AlgoConnectionInstance>> connInstanceLists
= groupedMassProblems.get(entry.getKey());

createProbability.put((AlgoMassProblem)entry.getKey(),
calculateMassProblemCreateProbability(connInstanceLists,
groupedConnectionInstanceMap));

destroyProbability.put((AlgoMassProblem)entry.getKey(),
calculateMassProblemDestroyProbability(connInstanceLists,
groupedConnectionInstanceMap));
        }
    }

// метод підліку ймовірності появи зв'язку

    private Map<AlgoConnection, Double>
calculateMassProblemCreateProbability(List<List<AlgoConnectionIn
stance>> connInstanceLists,
```

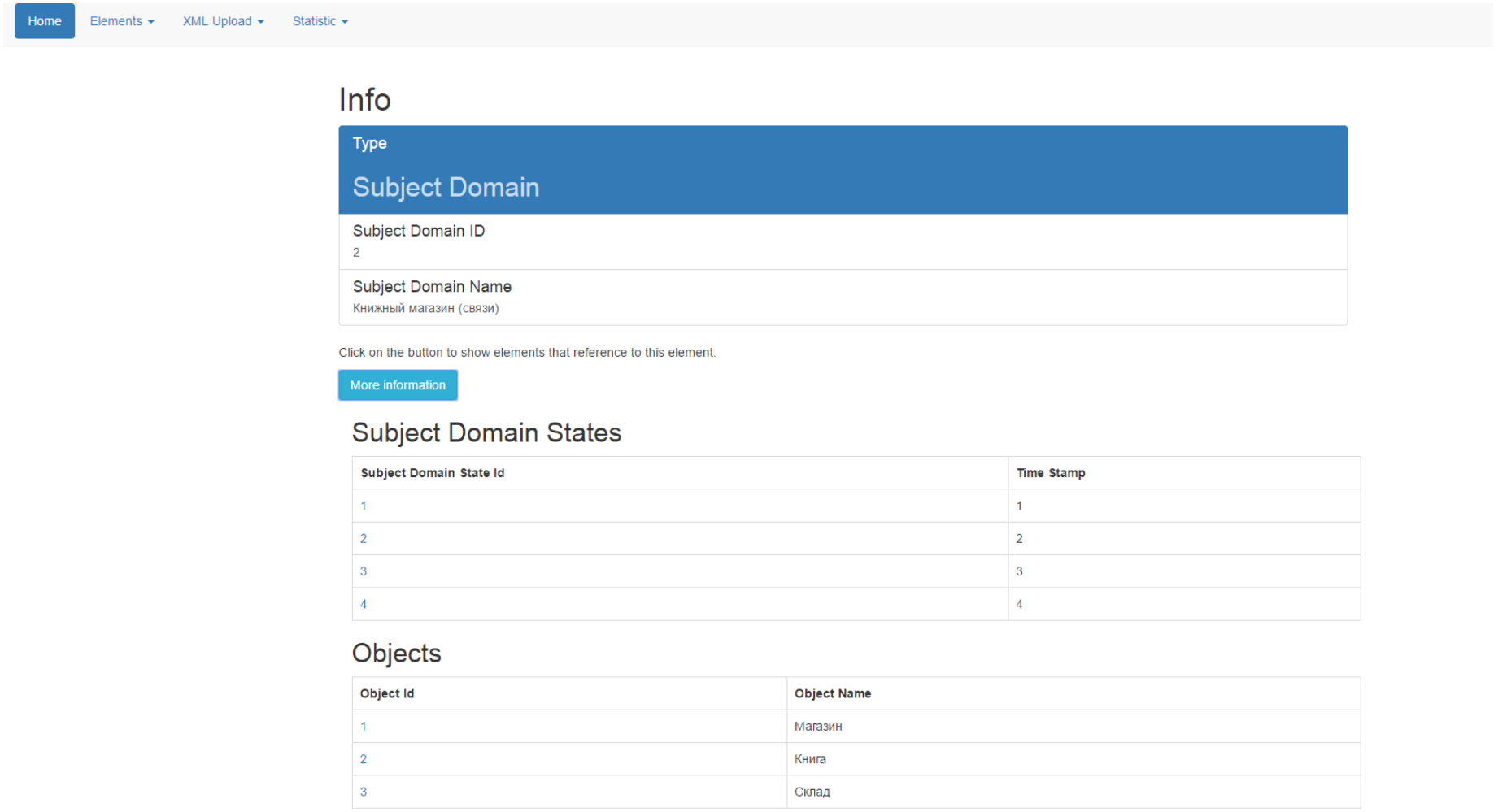
```
Map<AlgoConnection, Set<AlgoConnectionInstance>>
connConnectionInstanceSet){
        Map<AlgoConnection, Double> connectionValueMap = new
LinkedHashMap<AlgoConnection, Double>();

        for(Map.Entry entry :
connConnectionInstanceSet.entrySet()){
            int j = 0, k = 0;
            for(AlgoConnectionInstance connInstance :
(Set<AlgoConnectionInstance>)entry.getValue()){
                int m = 0, n = 0;
                boolean repeatFlag = false;
                for(int i = 0; i < connInstanceLists.size();
i++){

if(connInstanceLists.get(i).contains(connInstance)){
                        if(!repeatFlag){
                            if(i != 0){
                                m++;
                            }
                        }
                        repeatFlag = true;
                    } else repeatFlag = false;
                    n++;
                }
                j += m;
                k += n;
            }


connectionValueMap.put((AlgoConnection)entry.getKey(),
(double)j/k);
        }

        return connectionValueMap;
    }

// метод підліку ймовірності пропажі зв’язку

    private Map<AlgoConnection, Double>
calculateMassProblemDestroyProbability(List<List<AlgoConnectionI
nstance>> connInstanceLists,

Map<AlgoConnection, Set<AlgoConnectionInstance>>
connConnectionInstanceSet) {
        Map<AlgoConnection, Double> connectionValueMap = new
LinkedHashMap<AlgoConnection, Double>();

        for(Map.Entry entry :
connConnectionInstanceSet.entrySet()){
            int j = 0, k = 0;
```

```
            for(AlgoConnectionInstance connInstance :
(Set<AlgoConnectionInstance>)entry.getValue()){
                int m = 0, n = 0;
                boolean repeatFlag = false;
                for(int i = 0; i < connInstanceLists.size();
i++){

if(connInstanceLists.get(i).contains(connInstance)){
                        repeatFlag = true;
                    } else {
                        if(repeatFlag){
                            if(i != 0){
                                m++;
                            }
                            repeatFlag = false;
                        }
                    }
                    n++;
                }
                j += m;
                k += n;
            }


connectionValueMap.put((AlgoConnection)entry.getKey(),
(double)j/k);
        }

        return connectionValueMap;
    }

    public Map<AlgoConnection, Double> getStability(){
        return stability;
    }

    public Map<AlgoConnection, Double> getInevitability(){
        return inevitability;
    }

    public Map<AlgoMassProblem, Map<AlgoConnection, Double>>
getCreateProbability(){
        return createProbability;
    }

    public Map<AlgoMassProblem, Map<AlgoConnection, Double>>
getDestroyProbability(){
        return destroyProbability;
    }
```



ДОДАТОК Г ГОЛОВНА СТОРІНКА

ДОДАТОК Д СПИСОК ПРЕДМЕТНИХ ОБЛАСТЕЙ


Home
Elements
XML Upload
Statistic
Info
Type
Subject Domain
Subject Domain ID
2
Subject Domain Name
Книжный магазин (связи)
Click on the button to show elements that reference to this element.
More information
Subject Domain States
Subject Domain State Id
Time Stamp
1
1
2
2
3
3
4
4
Objects
Object Id
Object Name
1
Магазин
2
Книга
3
Склад

ДОДАТОК Ж ВИБІР ПРЕДМЕТНОЇ ОБЛАСТІ ДЛЯ АНАЛІЗА ЗА ДОПОМОГОЮ АЛГОРИТМУ ПОШУКУ ВИЗНАЧАЛЬНИХ АТРИБУТІВ

ДОДАТОК И
РЕЗУЛЬТАТИ РОБОТИ АЛГОРИТМУ ПОШУКУ ВИЗНАЧАЛЬНИХ АТРИБУТІВ

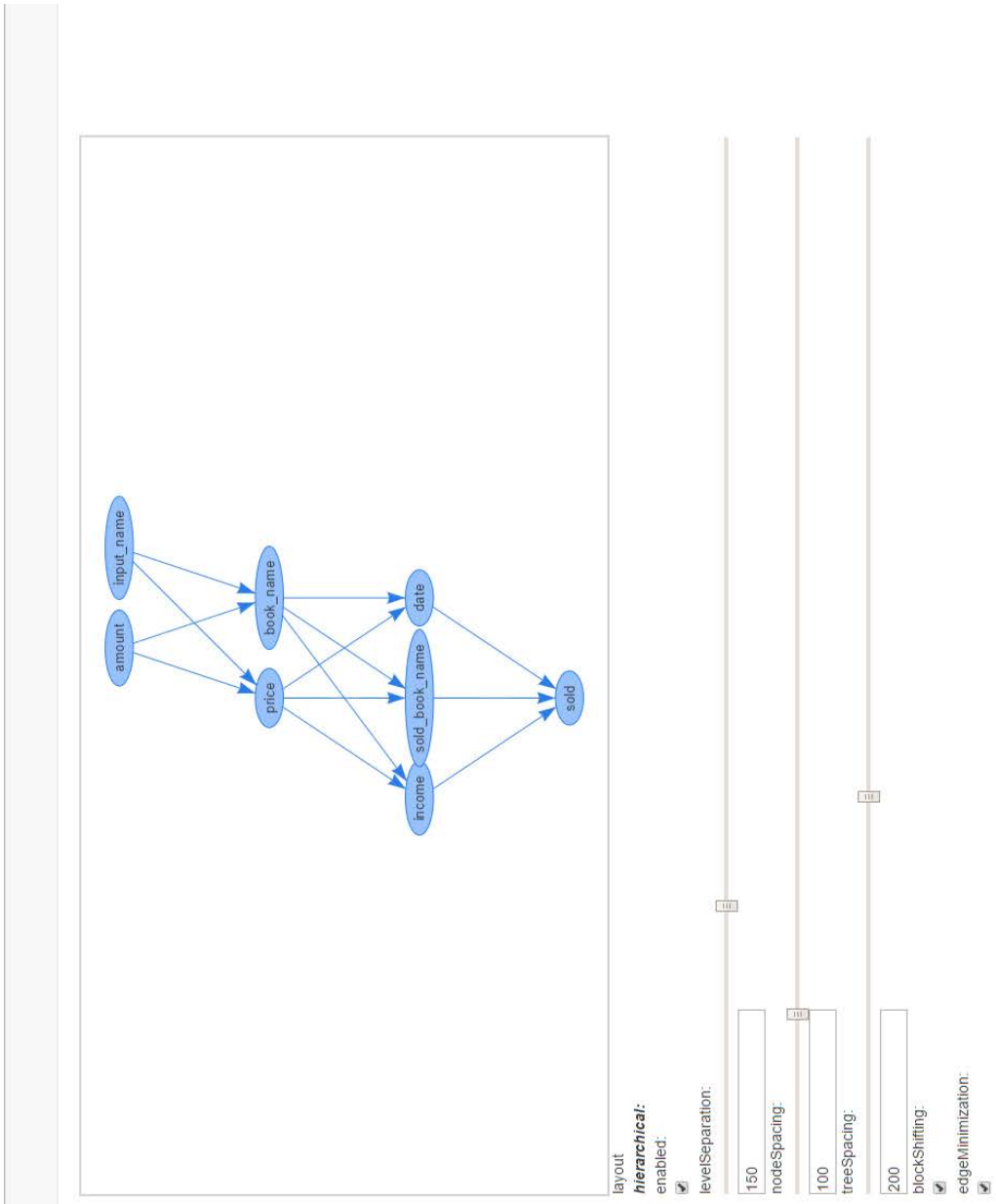


Рисунок И.1

Attribute	Measure
income	1.4
date	1.4
sold	1.0
sold_book_name	1.4
amount	4.2
input_name	4.2
price	3.0999999999999996
book_name	3.0999999999999996

Attribute Measures:

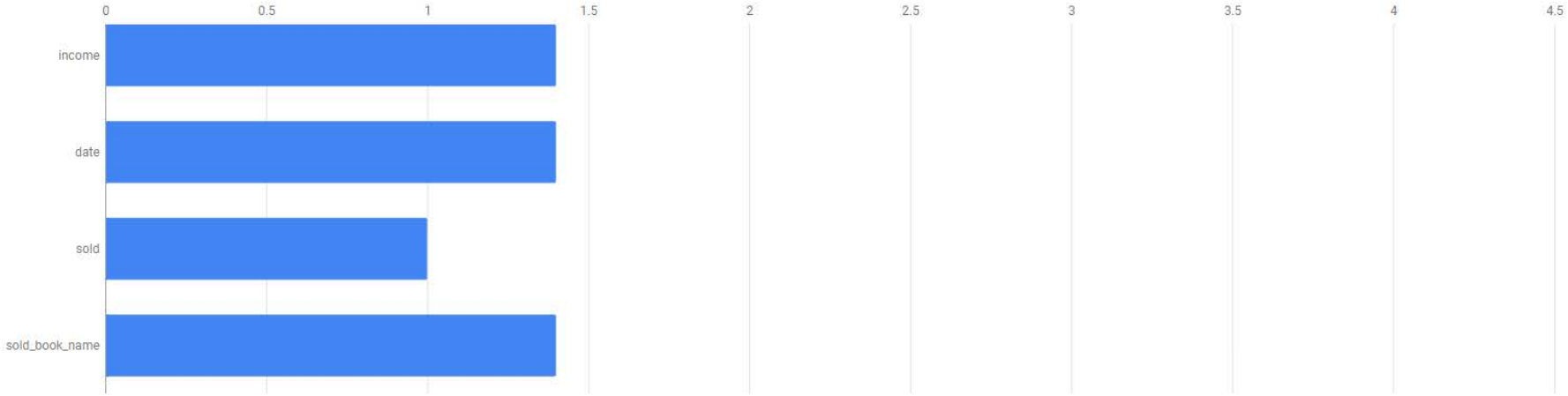
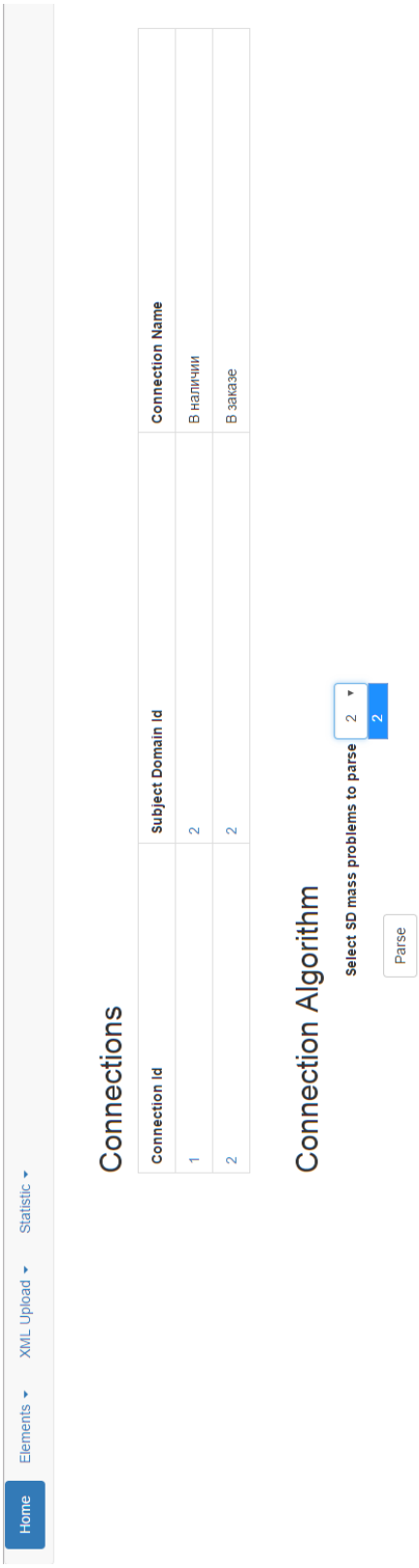


Рисунок И.2

ДОДАТОК К ВИБІР ПРО ДЛЯ АНАЛИЗА ЗА ДОПОМОГОЮ АЛГОРИТМУ КЛАСИФІКАЦІЇ ЗВ’ЯЗКІВ



ДОДАТОК Л РЕЗУЛЬТАТИ РОБОТИ АЛГОРИТМ КЛАСИФІКАЦІЇ ЗВ’ЯЗКІВ

Defining Attribute Algorithm Statistic

Attribute	Value
sold	1.0
date	1.39999998
income	1.39999998
sold_book_name	1.39999998
book_name	3.0999999
price	3.0999999
amount	4.19999981
input_name	4.19999981

Connection Algorithm Statistic

Attribute	Value	Connection Stat Type
Второй день работы/Книга - Склад	0.0	Create Probability
Второй день работы/Книга - Склад	0.5	Destroy Probability
Второй день работы/Магазин - Книга	0.25	Create Probability
Второй день работы/Магазин - Книга	0.0	Destroy Probability
Книга - Склад	0.666666687	Inevitability
Книга - Склад	0.333333343	Stability
Магазин - Книга	0.600000024	Inevitability
Магазин - Книга	0.400000006	Stability
Первый день работы/Книга - Склад	0.0	Create Probability
Первый день работы/Книга - Склад	0.0	Destroy Probability
Первый день работы/Магазин - Книга	0.0	Create Probability