ДОДАТОК А

Програмний код

```
package WEKA.classifiers.functions;

import lombok.*;
import org.apache.commons.io.output.*;
import org.apache.commons.lang3.time.StopWatch;
import org.deeplearning4j.*;
import WEKA.classifiers.*;
import WEKA.core.*;
import WEKA.dl4j.*;
import WEKA.filters.*;


/**
 * A wrapper for DeepLearning4j that can be used to train a multi-
layer perceptron.
 */
@Log4j2
@Data
public class Dl4jMlpClassifier extends RandomizableClassifier
                  implements    BatchPredictor,    CapabilitiesHandler,
IterativeClassifier {
    /**
     * The ID used for serializing this class.
     */
    private static final long serialVersionUID = -6363254116597574265L;
    /**
     * filter: Normalize training data
     */
    public static final int FILTER_NORMALIZE = 0;
    /**
     * filter: Standardize training data
     */
    public static final int FILTER_STANDARDIZE = 1;
    /**
     * filter: No normalization/standardization
     */
    public static final int FILTER_NONE = 2;
    /**
     * The filter to apply to the training data
     */
    public static final Tag[] TAGS_FILTER = {
        new Tag(FILTER_NORMALIZE, "Normalize training data"),
        new Tag(FILTER_STANDARDIZE, "Standardize training data"),
        new Tag(FILTER_NONE, "No normalization/standardization"),
    };
    /**
     * Filter used to replace missing values.
     */
    protected ReplaceMissingValues replaceMissingFilter;
    /**
     * Filter used to normalize or standardize the data.
     */
    protected Filter filter;
    /**
      * Filter used to convert nominal attributes to binary numeric
attributes.
     */
```

```
        protected NominalToBinary nominalToBinaryFilter;
        /**
         * ZeroR classifier, just in case we don't actually have any data
to train a network.
         */
        protected ZeroR zeroR;
        /**
         * The actual neural network model.
         */
        protected transient ComputationGraph model;
        /**
         * The model zoo model.
         */
        protected ZooModel zooModel = new CustomNet();
        /**
         * The size of the serialized network model in bytes.
         */
        protected long modelSize;
        /**
         * The layers of the network.
         */
           protected transient Layer[] layers = new Layer[]{new
OutputLayer()};
        /**
         * The configuration of the network.
         */
             protected NeuralNetConfiguration netConfig = new
NeuralNetConfiguration();
        /**
         * The configuration for early stopping.
         */
        protected EarlyStopping earlyStopping = new EarlyStopping();
        /**
         * The number of epochs to perform.
         */
        protected int numEpochs = 10;
        /** The current upper bound for the number of epochs */
        protected int maxEpochs = 0;
        /**
         * The total number of epochs that have been performed.
         */
        protected int numEpochsPerformed;

        /** The number of epochs performed in this session of iterating */
        protected int numEpochsPerformedThisSession;
        /**
         * The dataset trainIterator.
         */
        protected transient DataSetIterator trainIterator;
        /**
         * The training instances (set to null when done() is called).
         */
        protected Instances trainData;
        /**
         * The instance iterator to use.
         */
          protected AbstractInstanceIterator instanceIterator = new
DefaultInstanceIterator();
```

```
        /**
                * Queue size for AsyncDataSetIterator (if < 1,
AsyncDataSetIterator is not used)
         */
        protected int queueSize = 0;
        /**
         * Whether to normalize/standardize/neither
         */
        protected int filterType = FILTER_STANDARDIZE;
        /**
         * Coefficient x0 used for normalizing the class
         */
        protected double x0 = 0.0;
        /**
         * Coefficient x1 used for normalizing the class
         */
        protected double x1 = 1.0;
        /**
         * Caching mode to use for loading data
         */
        protected CacheMode cacheMode = CacheMode.MEMORY;
        /**
         * Training listener list
         */
        protected TrainingListener iterationListener = new EpochListener();
        /**
         * Flag indicating if initialization is finished.
         */
        protected boolean isInitializationFinished = false;
        /**
         * List of indices to store the label order.
         */
        protected int[] labelSortIndex;
        /**
         * Logging configuration.
         */
        protected LogConfiguration logConfig = new LogConfiguration();
        /**
         * Whether to allow training to continue at a later point after the
initial
         * model is built.
         */
        protected boolean resume;
        /**
         * The main method for running this class.
         *
         * @param argv the command-line arguments
         */
        public static void main(String[] argv) {
          runClassifier(new Dl4jMlpClassifier(), argv);
        }

        /**
        /**
         * Custom serialization method.
         *
         * @param oos the object output stream
         */
```

```
        private void writeObject(ObjectOutputStream oos) throws IOException
{
          // figure out size of the written network
               CountingOutputStream  cos  =  new  CountingOutputStream(new
NullOutputStream());
          if (isInitializationFinished) {
            ModelSerializer.writeModel(model, cos, false);
          }
          modelSize = cos.getByteCount();

          // default serialization
          oos.defaultWriteObject();

          // Write layer configurations
          String[] layerConfigs = new String[layers.length];
          for (int i = 0; i < layers.length; i++) {
               layerConfigs[i]  =  layers[i].getClass().getName()  +  "::"  +
WEKA.core.Utils
                .joinOptions(layers[i].getOptions());
          }
          oos.writeObject(layerConfigs);

          // actually write the network
          if (isInitializationFinished) {
            ModelSerializer.writeModel(model, oos, false);
          }
        }
        /**
         * Check if a given layer is a convolutional/subsampling layer
         *
         * @param layer Layer to check
         * @return True if layer is convolutional/subsampling
         */
        protected boolean isNDLayer(Layer layer) {
            return  layer  instanceof  ConvolutionLayer  ||  layer  instanceof
SubsamplingLayer;
        }

        /**
           * Check  if  layer  names  are  duplicate.  If  so,  correct  them  by
appending indices
         *
         * @param layers Array of network layer
         */
        protected void fixDuplicateLayerNames(Layer[] layers) {
                                          Set<String>         names         =
Arrays.stream(layers).map(Layer::getLayerName).collect(Collectors.toSet());

          for (String name : names) {
            // Find duplicates with the same name
            List<Layer> duplicates =
                Arrays.stream(layers)
                    .filter(l -> name.equals(l.getLayerName()))
                    .collect(Collectors.toList());

            // If no duplicates were found, continue
            if (duplicates.size() == 1) {
              continue;
```

```
            }

            // For each duplicate add an index
            for (int i = 0; i < duplicates.size(); i++) {
              duplicates.get(i).setLayerName(name + " " + (i + 1));
            }
          }
        }
        /**
         * Store the label sort index for mapping WEKA-labels to resorted
dl4j-labels.
         *
         * @param data Input data
         */
        protected void saveLabelSortIndex(Instances data) {
          if (data.classAttribute().isNominal()) {
            // Save Label order as DL4J automatically sorts them
            List<String> labels = new ArrayList<>();
            int numClassValues = data.classAttribute().numValues();
            for (int i = 0; i < numClassValues; i++) {
              labels.add(data.classAttribute().value(i));
            }
            List<String> labelsSorted = new ArrayList<>(labels);
            Collections.sort(labelsSorted);
            labelSortIndex = new int[numClassValues];

            for (int i = 0; i < numClassValues; i++) {
              String label = labels.get(i);
              int sortedIndex = labelsSorted.indexOf(label);
              labelSortIndex[i] = sortedIndex;
            }
          }
        }

        /**
         * Initialize early stopping with the given data
         *
         * @param data Data
         * @return Augmented data - if early stopping applies, return train
set without validation set
         */
          protected Instances initEarlyStopping(Instances data) throws
Exception {
          // Split train/validation
          double valSplit = earlyStopping.getValidationSetPercentage();
          Instances trainData;
          Instances valData;
          if (useEarlyStopping()) {
            // Split in train and validation
            Instances[] insts = splitTrainVal(data, valSplit);
            trainData = insts[0];
            valData = insts[1];
            validateSplit(trainData, valData);
                DataSetIterator valIterator = getDataSetIterator(valData,
cacheMode, "val");
            earlyStopping.init(valIterator);
          } else {
            // Keep the full data
```

```
            trainData = data;
          }

          return trainData;
        }

        /**
         * Validate a data split of train and validation data
         *
         * @param trainData Training data
         * @param valData Validation data
         * @throws WEKAException Invalid validation split
         */
          protected void validateSplit(Instances trainData, Instances
valData) throws WEKAException {
          if (earlyStopping.getValidationSetPercentage() < 10e-8) {
            // Use no validation set at all
            return;
          }
          int classIndex = trainData.classIndex();
                              int      valDataNumDinstinctClassValues      =
valData.numDistinctValues(classIndex);
                             int      trainDataNumDistinctClassValues      =
trainData.numDistinctValues(classIndex);
          if (trainData.numClasses() > 1
                                    &&    valDataNumDinstinctClassValues    !=
trainDataNumDistinctClassValues) {
            throw new InvalidValidationPercentageException(
                  "The validation data did not contain the same classes as
the training data. "
                    + "You should increase the validation percentage in the
EarlyStopping configuration.");
          }
        }
        /**
         * Apply WEKA filter preprocessing to the input data.
         *
         * @param data Data as WEKA instances
         * @return Preprocessed instances
         * @throws Exception Preprocessing failed
         */
          protected Instances preProcessInput(Instances data) throws
Exception {
          // Remove instances with missing class and check that instances
and
          // predictor attributes remain.
          data = new Instances(data);
          data.deleteWithMissingClass();
          zeroR = null;
          int numSamples = data.numInstances();
          if (numSamples == 0 || data.numAttributes() < 2) {
            zeroR = new ZeroR();
            zeroR.buildClassifier(data);
            return null;
          }

          // Retrieve two different class values used to determine filter
          // transformation
```

```
          double y0 = data.instance(0).classValue();
          int index = 1;
          while (index < numSamples && data.instance(index).classValue() ==
y0) {
            index++;
          }
          if (index == numSamples) {
            // degenerate case, all class values are equal
            // we don't want to deal with this, too much hassle
            throw new Exception(
                 "All class values are the same. At least two class values
should be different");
          }
          double y1 = data.instance(index).classValue();

          // Init and apply the filters
          data = initFilters(data);

          double z0 = data.instance(0).classValue();
          double z1 = data.instance(index).classValue();
          x1 = (y0 - y1) / (z0 - z1); // no division by zero, since y0 !=
y1
          // guaranteed => z0 != z1 ???
          x0 = (y0 - x1 * z0); // = y1 - x1 * z1

          // Randomize the data, just in case
          Random rand = new Random(getSeed());
          data.randomize(rand);
          return data;
        }

        /**
         * Initialize {@link ReplaceMissingValues}, {@link NominalToBinary}
and {@link Standardize} or
         * {@link Normalize} filters
         *
         * @param data Input data to set the input formal of the filters
         * @return Transformed data
         * @throws Exception Filter can not be initialized
         */
        protected Instances initFilters(Instances data) throws Exception {
          // Replace missing values
          replaceMissingFilter = new ReplaceMissingValues();
          replaceMissingFilter.setInputFormat(data);
          data = Filter.useFilter(data, replaceMissingFilter);

          // Replace nominal attributes by binary numeric attributes.
          nominalToBinaryFilter = new NominalToBinary();
          nominalToBinaryFilter.setInputFormat(data);
          data = Filter.useFilter(data, nominalToBinaryFilter);

          // Standardize or normalize (as requested), including the class

          if (filterType == FILTER_STANDARDIZE) {
            filter = new Standardize();
            filter.setOptions(new String[]{"-unset-class-temporarily"});
            filter.setInputFormat(data);
            data = Filter.useFilter(data, filter);
```

```
          } else if (filterType == FILTER_NORMALIZE) {
            filter = new Normalize();
            filter.setOptions(new String[]{"-unset-class-temporarily"});
            filter.setInputFormat(data);
            data = Filter.useFilter(data, filter);
          } else {
            filter = null;
          }

          return data;
        }

        /**
         * Build the multilayer network defined by the networkconfiguration
and the list of layers.
         */
        protected void createModel() throws Exception {
          final INDArray features = getFirstBatchFeatures(trainData);
          ComputationGraphConfiguration.GraphBuilder gb =
              netConfig.builder().seed(getSeed()).graphBuilder();

          // Set ouput size
          final Layer lastLayer = layers[layers.length - 1];
          final int nOut = trainData.numClasses();
          if (lastLayer instanceof FeedForwardLayer) {
            ((FeedForwardLayer) lastLayer).setNOut(nOut);
          }

                                if      (getInstanceIterator()      instanceof
CnnTextEmbeddingInstanceIterator) {
            makeCnnTextLayerSetup(gb);
          } else {
            makeDefaultLayerSetup(gb);
          }

          gb.setInputTypes(InputType.inferInputType(features));
                                ComputationGraphConfiguration      conf      =
gb.pretrain(false).backprop(true).build();
          ComputationGraph model = new ComputationGraph(conf);
          model.init();
          this.model = model;
        }

        /**
         * Returns the activations at a certain
         *
         * @param layerName Layer name to get the activations from
         * @return Activations in form of instances
         */
        public Instances getActivationsAtLayer(String layerName, Instances
input) throws Exception {
          DataSetIterator iter = getDataSetIterator(input);
          iter.reset();
          DataSet next;
          INDArray acts = null;
          while (iter.hasNext()) {
            next = iter.next();
            INDArray features = next.getFeatures();
```

```
            int layerIdx = model.getLayer(layerName).getIndex() - 1;
             Map<String, INDArray> activations = model.feedForward(features,
layerIdx, false);
            INDArray activationAtLayer = activations.get(layerName);

            if (acts == null) {
              acts = activationAtLayer;
            } else {
              acts = Nd4j.concat(0, acts, activationAtLayer);
            }
          }

          if (acts == null) {
            return new Instances(input, 0);
          } else {
            return Utils.ndArrayToInstances(acts);
          }
        }
      }
```

ДОДАТОК Б

Слайди презентації

1

Об'єкт дослідження

процес контролю знань у дистанційній формі навчання

- Питання комп'ютерного контролю недостатньо широко освітлені в теоретичному плані, і інтерес до них зазвичай реалізується в більшості випадків шляхом створення чергової програми комп'ютерного контролю із заздалегідь складеним набором контрольних завдань.
- Засоби адаптивного тестування недостатньо освітлені й пророблені недостатньо прозоро.

2

Мета роботи

підвищення ефективності контролю знань суб'єктів навчання при дистанційній формі навчання за рахунок застосування адаптивних методів комп'ютерного тестування.

Для досягнення мети необхідно розв'язати наступні завдання.

- провести аналіз сучасного стану проблеми контролю знань у дистанційній формі навчання;
- провести аналіз існуючих систем комп'ютерного тестування;
- розробити адаптивний метод контролю знань суб'єктів навчання у дистанційній формі навчання;
- розробити прототип програмного забезпечення для адаптивного тестування в дистанційному навчанні.

3

Реалізація функцій контролю знань

- **Контролююча функція** полягає у виявленні стану знань і вмінь учнів, рівня їх розвитку, у вивченні ступеня засвоєння прийомів пізнавальної діяльності, навичок раціональної навчальної праці.
- **Навчальна функція** контролю полягає в удосконалюванні знань і вмінь, їх систематизації.
- **Діагностичні функції** контролю – одержання інформації про помилки, недоліки й пробіли у знаннях і вміннях учнів та причинах, що їх породжують, утруднень учнів в оволодінні навчальним матеріалом, про число помилок.
- **Прогностична функція** перевірки служить одержанню випереджальної інформації про навчально-виховний процес.
- **Розвиваюча функція** полягає в стимулюванні пізнавальної активності учнів, у розвитку їх творчих здібностей.
- **Контроль, що орієнтує**, – одержання інформації про ступінь досягнення мети навчання окремим учнем і групою в цілому – наскільки засвоєний і як глибоко вивчений навчальний матеріал.
- Функція **контролю, що виховує**, полягає у вихованні в учнів відповідального відношення до навчання, дисципліни, акуратності, чесності.

4

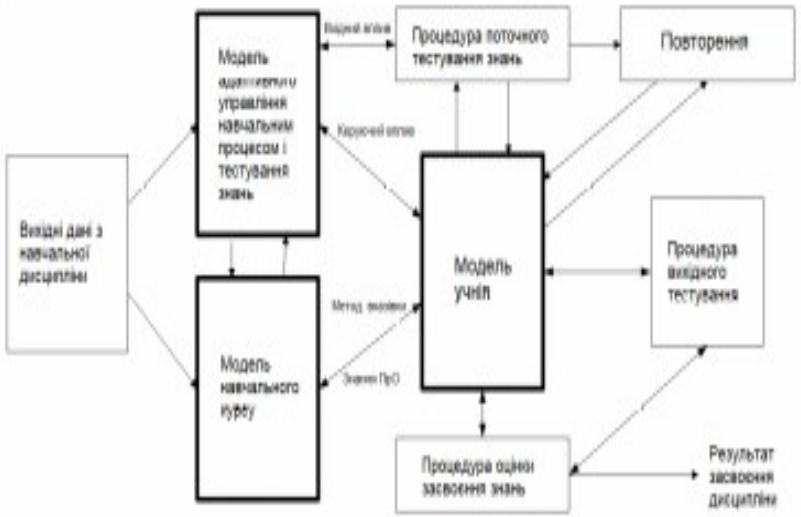
Адаптивний тестовий контроль

- Комп'ютеризована система науково обґрунтованої перевірки й оцінки результатів навчання, що має високу ефективність за рахунок оптимізації процедур генерації, пред'явлення й оцінки результатів виконання адаптивних тестів.
- Ефективність контрольно-оцінних процедур підвищується при використанні багатокрокової стратегії відбору й пред'явлення завдань, заснованої на алгоритмах з повною контекстною залежністю, у яких черговий крок відбувається тільки після оцінки результатів виконання попереднього кроку (або кроків).

Особливості адаптивного тестування:

- дозволяє більш гнучко й точно вимірювати знання тих, кого навчають;
- дозволяє вимірювати знання меншою кількістю завдань, ніж в класичній моделі;
- виявляє теми, які суб'єкт навчання, знає погано і система дозволяє задати по них низку додаткових питань;
- заздалегідь невідомо, скільки питань необхідно задати тому, кого тестують, щоб визначити його рівень знань;
- якщо питань, закладених у систему тестування, виявляється недостатньо, можна перервати тестування й оцінювати результат по тій кількості питань, на яку відповів той, кого навчають;
- можливе застосування тільки на ПК.

Взаємодія моделей і процесів адаптивної комп'ютеризованої системи навчання і тестування



6

Постановка задач дослідження

Комплексна модель адаптивної КСНТ повинна включати:

- семантичну модель навчального курсу;
- модель тих, кого навчають;
- модель організації процесу навчання й тестування знань тих, кого навчають.

Предметна область навчальної дисципліни, представлена моделлю знань навчальних матеріалів.

У якості моделі знань про структуру навчального матеріалу КСНТ обрана семантична мережа, яка містить відомості про поняття предметної області навчальної дисципліни, про їхні взаємозв'язки, складність окремих понять.

Під методичним матеріалом розуміють сукупність інформації, що містить знання про цілі, форми, методи й засобах навчання й рішення навчально-виховних завдань.

Модель суб'єкта навчання, вирішує завдання опису попередніх знань тих, кого навчають, про навчальний курс і їх індивідуальних особистісних характеристик і містить досить повну інформацію

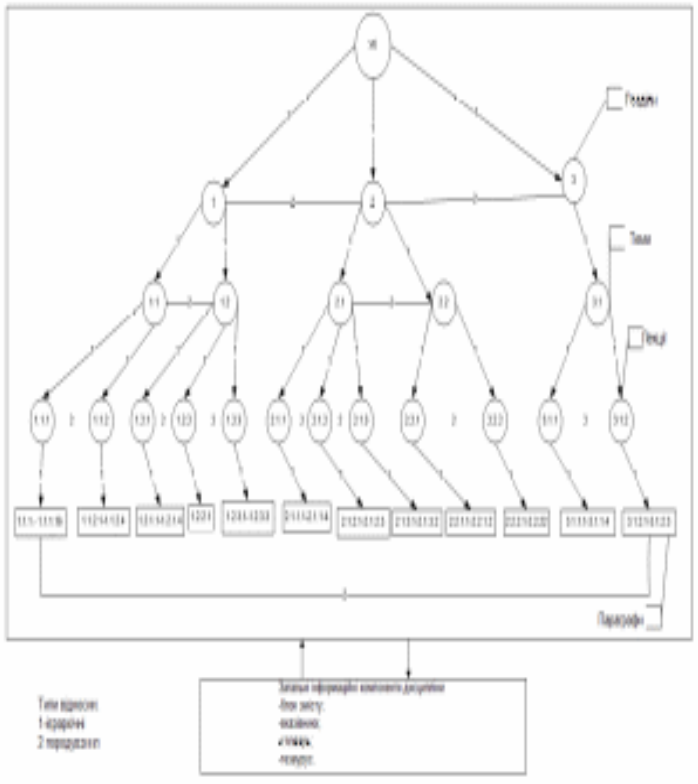
Алгоритм підбору й пред'явлення завдань будується за принципом зворотного зв'язку, коли при правильній відповіді випробуваного чергове завдання вибирається більш важким, а невірна відповідь спричиняє пред'явлення наступного більш легкого завдання, ніж те, на яке випробуваним була дана невірна відповідь.

Таким чином, у процесі тестування КСНТ адаптується до рівня знань суб'єкта навчання.

7

Модель дисципліни на базі семантичних мереж

- Модель навчального курсу на базі семантичної мережі реалізує адаптивне представлення інформації в базі навчальної дисципліни.
- Модель використовує строгий формальний апарат семантичних мереж, що дозволяє автоматизувати рішення (розробити алгоритм) низки фундаментальних завдань навчального процесу



8

Модель суб'єкта навчання

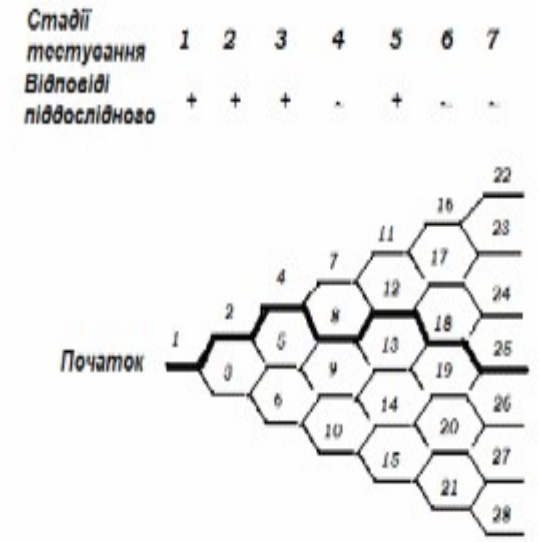
повинна містити в собі інформацію:

- про мету навчання;
- про знання того, якого навчають, у рамках досліджуваного курсу (поточний стан процесу навчання);
- про особливості подачі навчальних матеріалів і вибору контрольних завдань і питань;
- про правила зміни моделі того, якого навчають, за результатами роботи з тим, яких навчають,.

Моделі суб'єкта навчання, можуть бути розділені на дві основні групи: **фіксуючі й імітаційні**. Фіксуючі, у свою чергу, включають скалярні, оверлейні (векторні й мережні) і генетичні графи. До імітаційних можна віднести моделі обмежень, помилок і фальш-правил

9

Пірамідальна модель адаптивного тестування



10

Схема проходження завдань при постійній адаптації

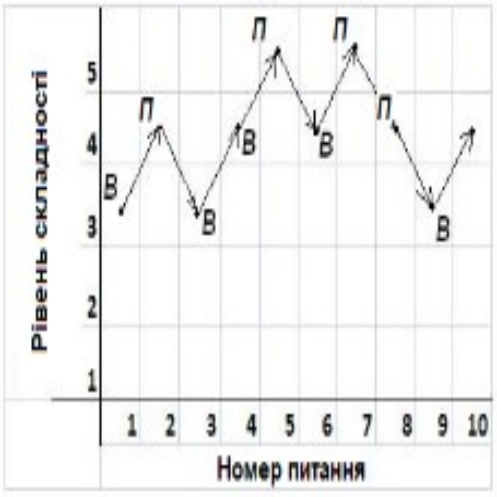


Схема проходження завдань при блоковій адаптації

11

Рішення цієї дилеми

- реалізація алгоритму з можливістю повернення назад, для того щоб поліпшити наступні рішення;
- реалізація маршового алгоритму, який би враховував наслідки прийнятих у майбутньому рішень.

В адаптивному тестуванні відстеження у зворотному порядку неможливо; алгоритм застосовується в масштабі реального часу й попередній вибір не може бути відмінний.
Таким чином, залишається тільки одна можливість: використовувати алгоритм, який щораз вибирає нове завдання.
Це новий клас алгоритмів

12

Опис методу роботи системи адаптивного тестування

- Сам алгоритм адаптивного тестування повинен мати на кожному кроці (при переході від одного завдання до іншого) максимальну інформативність (тобто надавати максимум інформації про здатності студента на одне задане питання).
- У той же час, не можна повністю відмовлятися від обліку суб'єктивних властивостей студента, які можуть виражатися в нерозумінні очевидно поставленого питання або завдання.

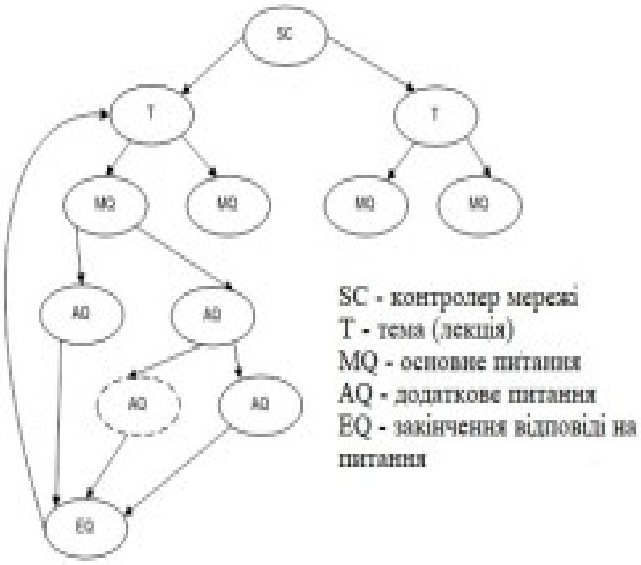
- Система тестів є замкненою в розумінні Марковских мереж, тобто якщо питання (основне або додаткове) являє собою стан *Si*, що має ймовірність перебування в ній системи P(si), то сумарна ймовірність P∑ перебування системи в S тотожно визначається як:

$$P\sum=\sum P(si)=1;$$

13

Модифікований алгоритм адаптивного тестування

Фрагмент мережі питань



Кращими є підходи, що засновані
на класі багатоступінчастих
методів адаптивного тестування,
які ефективно підтримують
комплексну модель адаптивного

Моделювання рівномірного розподілу питань

Обрана модель представлення тестових завдань із використанням додаткових питань на основі мереж Маркова, може забезпечити вирішення протиріччя між адекватною статистичною оцінкою знань і вмінь випробуваного й обраного методу, заснованого на принципі максимуму інформативності

Крок1. Визначається випадкове ціле число, що відповідає в загальному випадку кроку n-1 (позначено змінну, відповідну до цього числа як Wn-1).
Значення Wn-1 множиться на константу k1 і ділиться на іншу цілу константу p., тобто Wn-1:

$$Wn = k1 * Wn-1 (\mathrm{mod}\ p).$$

Залишок від розподілу є цілим числом, яке вибирається в якості випадкового цілого на кроці n

Прийнята анотація безлічі питань обумовлена відповідністю представлення нумерації масивів в обраній (C++) мові програмування.

Крок 2. Перевірка, $W_n \geq 2$ якщо $W_n < 2$ то $W_n = 2$

Крок 3. Здійснюється перевизначення Wn = Wn-1 і перехід до кроку 1, у випадку, якщо це за якимись причинами необхідно, інакше — закінчення роботи.

Рекомендоване значення P= 999563, k = 470 001.

Реалізація модифікованого алгоритму вибору тестових завдань

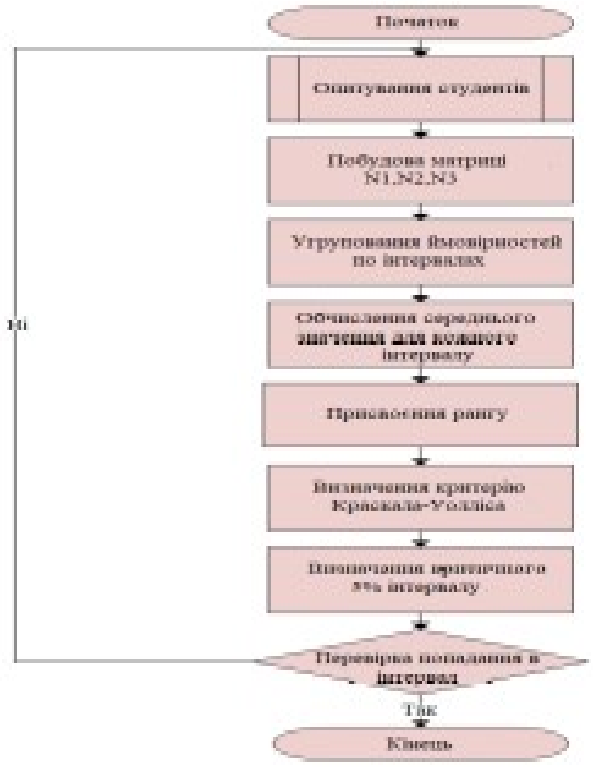
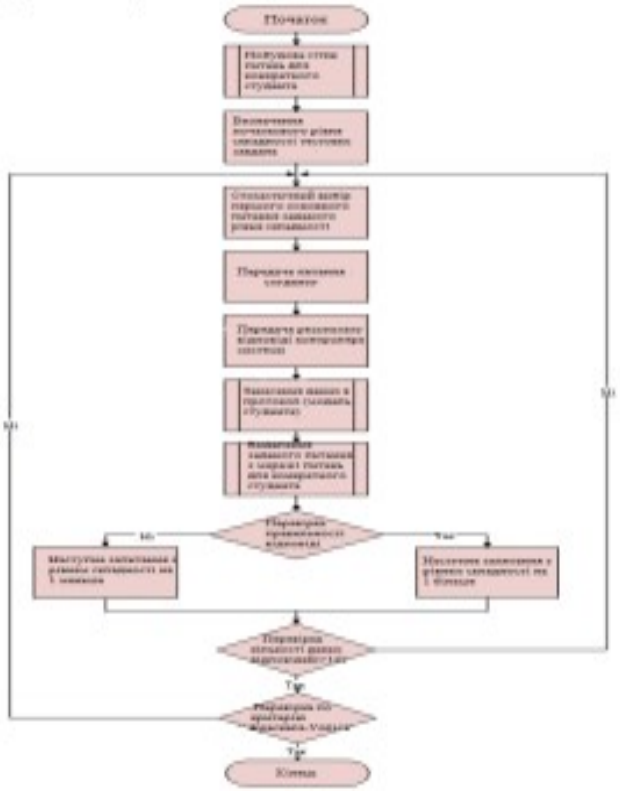


Схема алгоритму опитування

Пірамідальний алгоритм опитування студентів, реалізація якого відповідає моделі Г. Раша

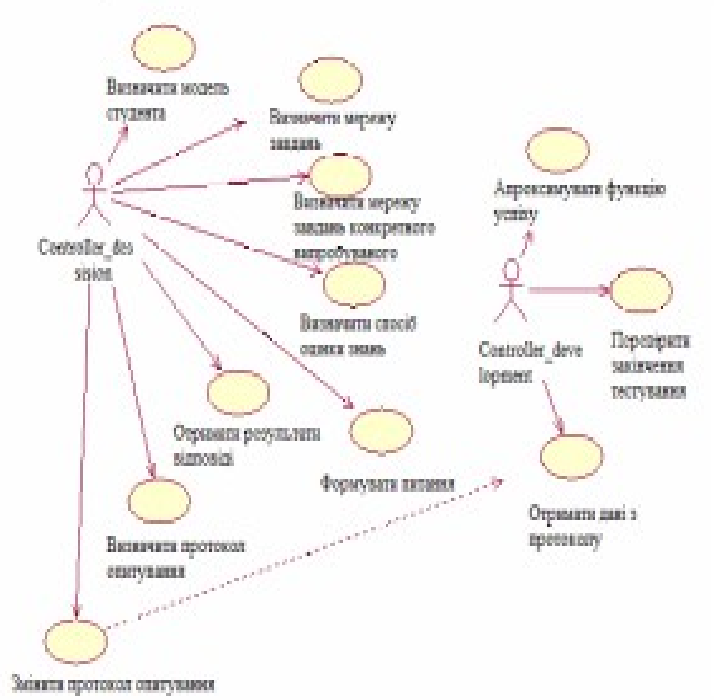


17

Діаграма варіантів використання

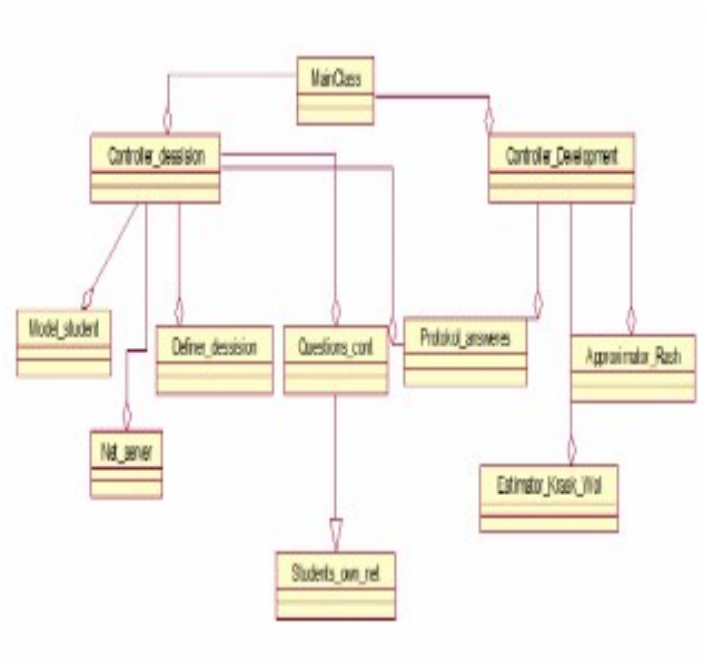
Розробка даного ПЗ зводиться до проектування моделі ПЗ чисельного експерименту, що включає статистичні методи обробки.

Вибір складових частин цієї моделі залежить від успішного застосування математичних методів, на яких вони засновані



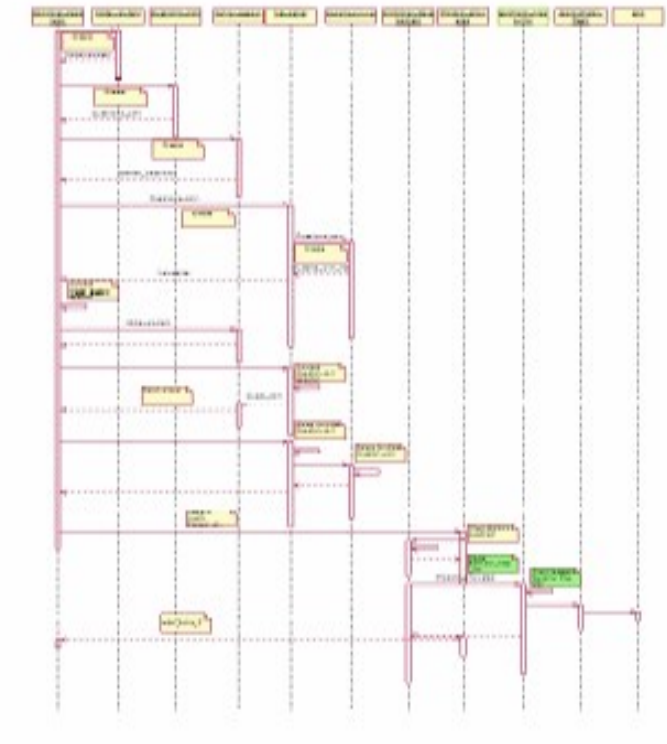
18

Діаграма класів



19

Діаграма послідовностей



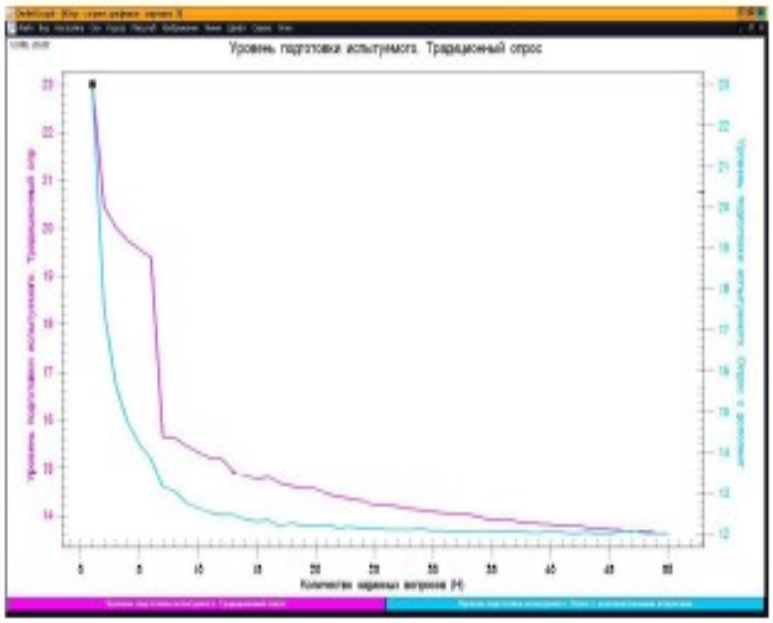
20

Інтерфейс проектованого прототипу ПЗ

Побудова мережі завдань із додатковими питаннями здійснюється розробленим ПЗ стохастичним способом, тобто припускає дослідження рішень на безлічі випадково побудованих семантичних мереж, що підкоряються правилам мереж Маркова: мережа повинна бути замкнена, не мати тупикових станів і визначати набір повних станів системи.

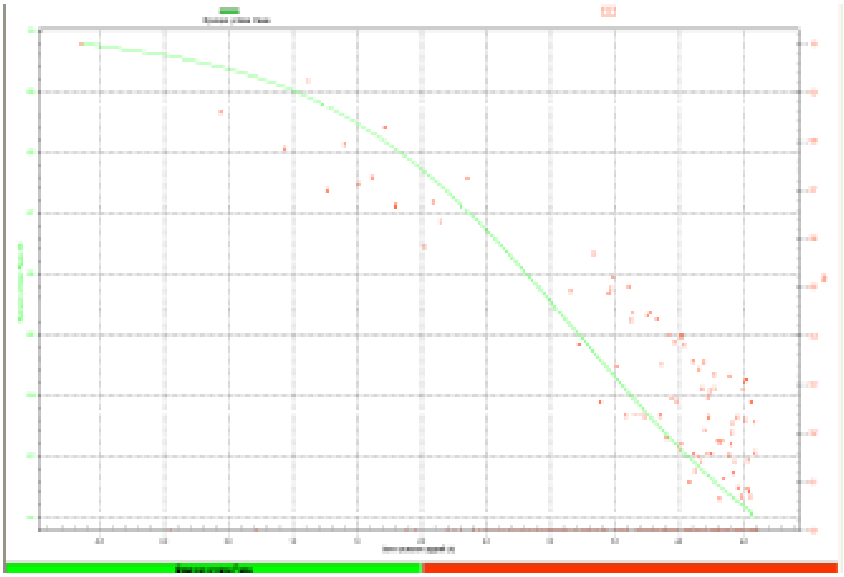
21

Графік рівнів підготовки випробуваного від авторитарного до латентного з використанням додаткових питань і без додаткових питань.



22

Екранна форма практичного результату апроксимації функції успіху від логіту складності одного випробуваного й експериментальні дані роботи програми.



ВИСНОВКИ

- В результаті виконання магістерської роботи показано перевагу запропонованого методу побудови ПЗ адаптивного тестування на базі мереж Маркова з елементами мереж Петрі при використанні концепції основних і додаткових питань.
- Різниця в рівнях складності основних і додаткових питань і здійснюваний зв'язок між основними і пілками додаткових питань дозволяє в процесі тестування як мінімізувати кількість необхідних відповідей тих, кого навчають, для визначення рівня їх знань, так і суттєво поліпшити адаптаційні властивості тестування.
- Результат по використанню зазначеної методики стосовно до 3-х параметричної моделі тестування Г. Раша довів, що апроксимація функції успіху, що припускає визначення уточнених значень дискримінаторів завдань шляхом рішення завдання оптимізації при використанні відомих алгоритмів визначило більші витрати машинного часу.
- Застосування запропонованих алгоритмів є ефективним для будь-яких обсягів і інших властивостей, що визначають тестування як процес

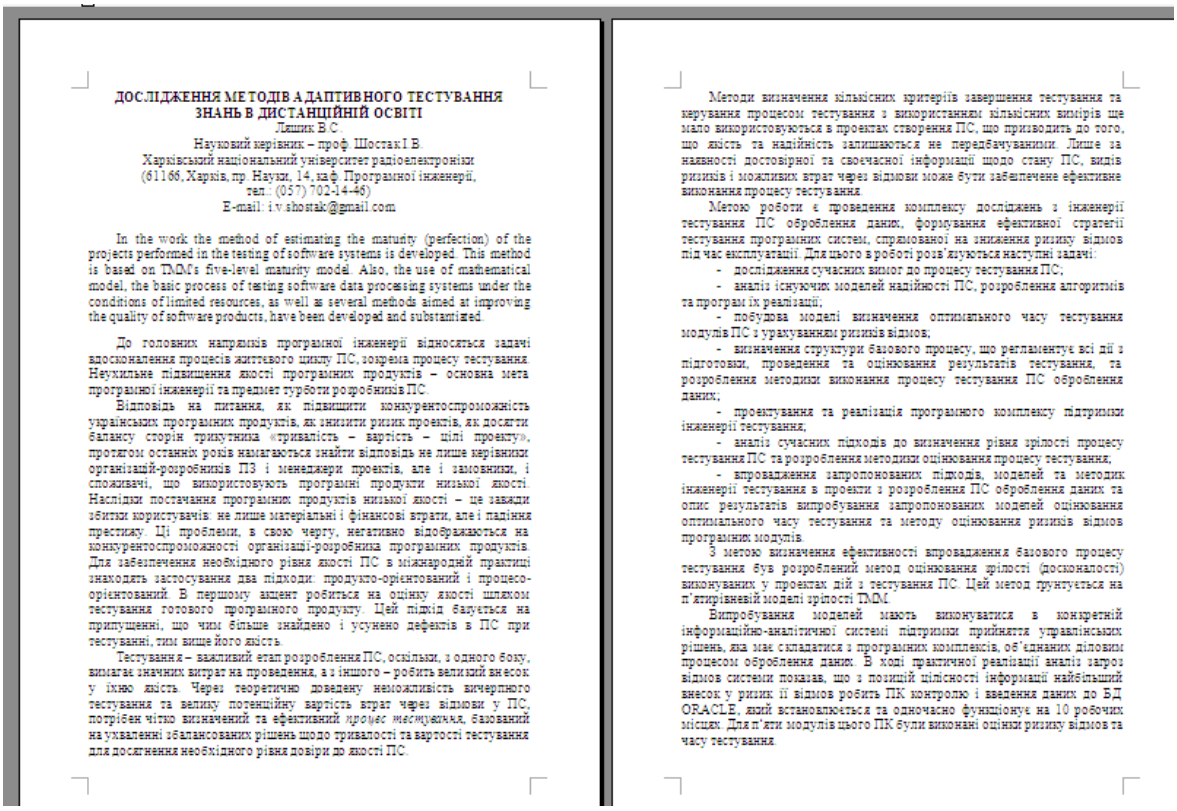
ВИСНОВКИ (продовження)

- Експериментальне підтвердження засноване на множині тестових завдань, що імітують умову проведення тестування при різних вихідних даних про апріорні властивості групи суб'єктів тестування.
- Створено програмне забезпечення, що забезпечує постановку чисельного експерименту на моделях тестування.
- Розроблене ПЗ протестовано на предмет працездатності й рішення поставленого завдання, тестування проводилося на контрольних прикладах з відкритих джерел і показало коректну працездатність розробленого ПЗ.
- На підставі проведення перерахованих вище тестів можна зробити висновок, що створений прототип виконує всі функції, зазначені у вихідних даних на розробку й відповідає всім пред'явленим до нього вимогам.

ДОДАТОК В

Апробація результатів роботи

Подано тези доповіді на Міжнародний молодіжний форум «Радіоелектроніка і молодь в XXI сторіччі



ДОСЛІДЖЕННЯ МЕТОДІВ АДАПТИВНОГО ТЕСТУВАННЯ ЗНАНЬ В ДИСТАНЦІЙНІЙ ОСВІТІ

Ляшик В.С.
Науковий керівник – проф. Шостак І.В.
Харківський національний університет радіоелектроніки
(61166, Харків, пр. Науки, 14, каф. Програмної інженерії,
тел.: (057) 702-14-46)
E-mail: i.v.shostak@gmail.com

In the work the method of estimating the maturity (perfection) of the projects performed in the testing of software systems is developed. This method is based on TMM's five-level maturity model. Also, the use of mathematical model, the basic process of testing software data processing systems under the conditions of limited resources, as well as several methods aimed at improving the quality of software products, have been developed and substantiated.

До головних напрямків програмної інженерії відносяться задачі вдосконалення процесів життєвого циклу ПС, зокрема процесу тестування. Неухильне підвищення якості програмних продуктів – основна мета програмної інженерії та предмет турботи розробників ПС.

Відповідь на питання, як підвищити конкурентоспроможність українських програмних продуктів, як знизити ризик проектів, як досягти балансу сторін трикутника «тривалість – вартість – цілі проекту», протягом останніх років намагаються знайти відповідь не лише керівники організацій-розробників ПЗ і менеджери проектів, але і замовники, і споживачі, що використовують програмні продукти низької якості. Наслідки постачання програмних продуктів низької якості – це завжди збитки користувачів: не лише матеріальні і фінансові втрати, але і падіння престижу. Ці проблеми, в свою чергу, негативно відображаються на конкурентоспроможності організації-розробника програмних продуктів. Для забезпечення необхідного рівня якості ПС в міжнародній практиці знаходять застосування два підходи: продукто-орієнтований і процесо-орієнтований. В першому акцент робиться на оцінку якості шляхом тестування готового програмного продукту. Цей підхід базується на припущенні, що чим більше знайдено і усунено дефектів в ПС при тестуванні, тим вище його якість.

Тестування – важливий етап розроблення ПС, оскільки, з одного боку, вимагає значних витрат на проведення, а з іншого – робить великий внесок у їхню якість. Через теоретично доведену неможливість вичерпного тестування та велику потенційну вартість втрат через відмови у ПС, потрібен чітко визначений та ефективний *процес тестування*, базований на ухваленні збалансованих рішень щодо тривалості та вартості тестування для досягнення необхідного рівня довіри до якості ПС.

Методи визначення кількісних критеріїв завершення тестування та керування процесом тестування з використанням кількісних вимірів ще мало використовуються в проектах створення ПС, що призводить до того, що якість та надійність залишаються не передбачуваними. Лише за наявності достовірної та своєчасної інформації щодо стану ПС, видів ризиків і можливих втрат через відмови може бути забезпечене ефективне виконання процесу тестування.

Метою роботи є проведення комплексу досліджень з інженерії тестування ПС оброблення даних, формування ефективної стратегії тестування програмних систем, спрямованої на зниження ризику відмов під час експлуатації. Для цього в роботі розв'язуються наступні задачі:

- дослідження сучасних вимог до процесу тестування ПС;
- аналіз існуючих моделей надійності ПС, розроблення алгоритмів та програм їх реалізації;
- побудова моделі визначення оптимального часу тестування модулів ПС з урахуванням ризиків відмов;
- визначення структури базового процесу, що регламентує всі дії з підготовки, проведення та оцінювання результатів тестування, та розроблення методики виконання процесу тестування ПС оброблення даних;
- проектування та реалізація програмного комплексу підтримки інженерії тестування;
- аналіз сучасних підходів до визначення рівня зрілості процесу тестування ПС та розроблення методики оцінювання процесу тестування;
- впровадження запропонованих підходів, моделей та методик інженерії тестування в проекти з розроблення ПС оброблення даних та опис результатів випробування запропонованих моделей оцінювання оптимального часу тестування та методу оцінювання ризиків відмов програмних модулів.

З метою визначення ефективності впровадження базового процесу тестування був розроблений метод оцінювання зрілості (досконалості) виконуваних у проектах дій з тестування ПС. Цей метод ґрунтується на п'ятирівневій моделі зрілості TMM.

Випробування моделей мають виконуватися в конкретній інформаційно-аналітичної системі підтримки прийняття управлінських рішень, яка має складатися з програмних комплексів, об'єднаних діловим процесом оброблення даних. В ході практичної реалізації аналіз загроз відмов системи показав, що з позицій цілісності інформації найбільший внесок у ризик її відмов робить ПК контролю і введення даних до БД ORACLE, який встановлюється та одночасно функціонує на 10 робочих місцях. Для п'яти модулів цього ПК були виконані оцінки ризику відмов та часу тестування.