УДК 004.43, 004.9

© 2011 р. Г.Г. Киричек, О.О. Киричек

Запорізький національний технічний університет, Запоріжжя
Національний технічний університет України «КПІ», Київ

МОДЕЛЬ СИСТЕМИ ОЦІНКИ ІДЕНТИЧНОСТІ ПРОГРАМНОГО КОДУ

Запропоновано модель системи оцінки ідентичності програмного коду для використання при вивченні дисциплін з програмування. Дослідження присвячено вирішенню актуального завдання підвищення ефективності вивчення і практичного застосування мов програмування в процесі навчання. Система забезпечує аналіз та обробку результатів розробки програмних проектів, виконаних студентами та визначення випадків повторного використання коду.

The model of program code identity estimation system has been proposed. The given model is devoted to being used during the process of studying programming disciplines. The investigation is aimed at the increasing of programming languages studying and practical implementation effectiveness during studying process which is the actual task. The system provides the analysis and processing of students' software projects engineering results and checking the occurrences of re-usable code.

Вступ

Рівень досягнень багатьох українських вищих навчальних закладів (ВНЗ) в галузі інформатизації навчального процесу та використанні новітніх методів контролю знань є достатньо високим [1]. Але зараз, у час інтенсивного розвитку інформаційних технологій інтелектуальна власність стає більш цінною. Можливість легкого копіювання інформації, що представлена у електронному вигляді, навіть при виконанні індивідуальних завдань з розробки програмного коду при вивченні дисциплін з програмування, породила багато проблем, що пов'язані з порушенням авторських прав. У зв'язку з цим виникла необхідність у потужних інструментах для захисту авторських прав при вивченні дисциплін з програмування.

Задачею системи оцінки ідентичності програмного коду є автоматичне виявлення (за заданими критеріями) того, чи була використана у програмі чужа ідея. На практиці певним чином задаються функція близькості і поріг, за якими можна визначити наскільки ймовірно, що певна частина програмного коду була запозичена [2].

Актуальність та доцільність даної розробки полягає у необхідності визначення повторного використання програмного коду у творчих проектах студентів, задля кращого закріплення ними практичних навичок. Завдяки впровадженню такої системи з'явиться інструмент, що надасть можливість викладачам циклу дисциплін з програмування більш точно оцінити практичні навички, що отримав студент під час виконання проектів.

1. Загальна структура системи

Система оцінки ідентичності програмного коду включає відпрацювання наступних функцій:

– реєстрацію користувачів, створення груп користувачів;

– імпорт проекту, пошук файлів по заданим критеріям;

– аналіз плагіату в структурі проекту (ідентичність структури каталогів та зображень);

– відстеження часу, дати, авторів та внесення змін до програми;

– аналіз програмного коду в залежності від типу файлу.

Впровадження системи має ряд цілей:

– контролювати повторне використання студентами коду проектів;

– створення бази навчальних проектів.

Призначення системи полягає у використанні її для аналізу творчих проектів студентів, а саме при розробці сайтів, ігор, або специфічних програм.

Система повинна задовольняти наступним вимогам:

– забезпечення користувачам зручного та зрозумілого системного інтерфейсу;

– можливість перегляду документів що аналізуються;

– внесення інформації до бази даних (БД) про створення нових елементів системи (груп, користувачів, типів файлів, проектів т.ін.);

– забезпечення легкої подальшої адаптації та доробки системи.

Вона складається з таких груп функцій:

– збір даних;

– введення/виведення оброблених даних до відповідної системи керування базами даних (СКБД);

– виведення результатів аналізу у вигляді звітних форм.

Система керування базами даних забезпечує:

– моделювання даних предметної області (визначення структур баз даних і маніпуляцію даними);

– підтримку цілісності БД та її відновлення після перебоїв;

– обмін даними з клієнтами, підключеними до сервера;

– управління транзакціями;

– імпорт/експорт інформації;

– архівування/відновлення даних;

– підтримку стандарту *OLEDB* і наявність власних інтерфейсів прикладних програм (*API*), сумісних з програмними засобами клієнтів локальних мереж.

2. Проектування системи

2.1. Проектування функціональної моделі

Для відображення функціональної моделі системи оцінки ідентичності програмного коду застосовано прецеденти (рис. 1).

Система накопичує дані про проект, який надійшов на аналіз. Користувачі мають можливість створювати групи, облікові записи, та імпортувати закріплені за ними проекти.

Також є можливість встановлювати параметри аналізу.

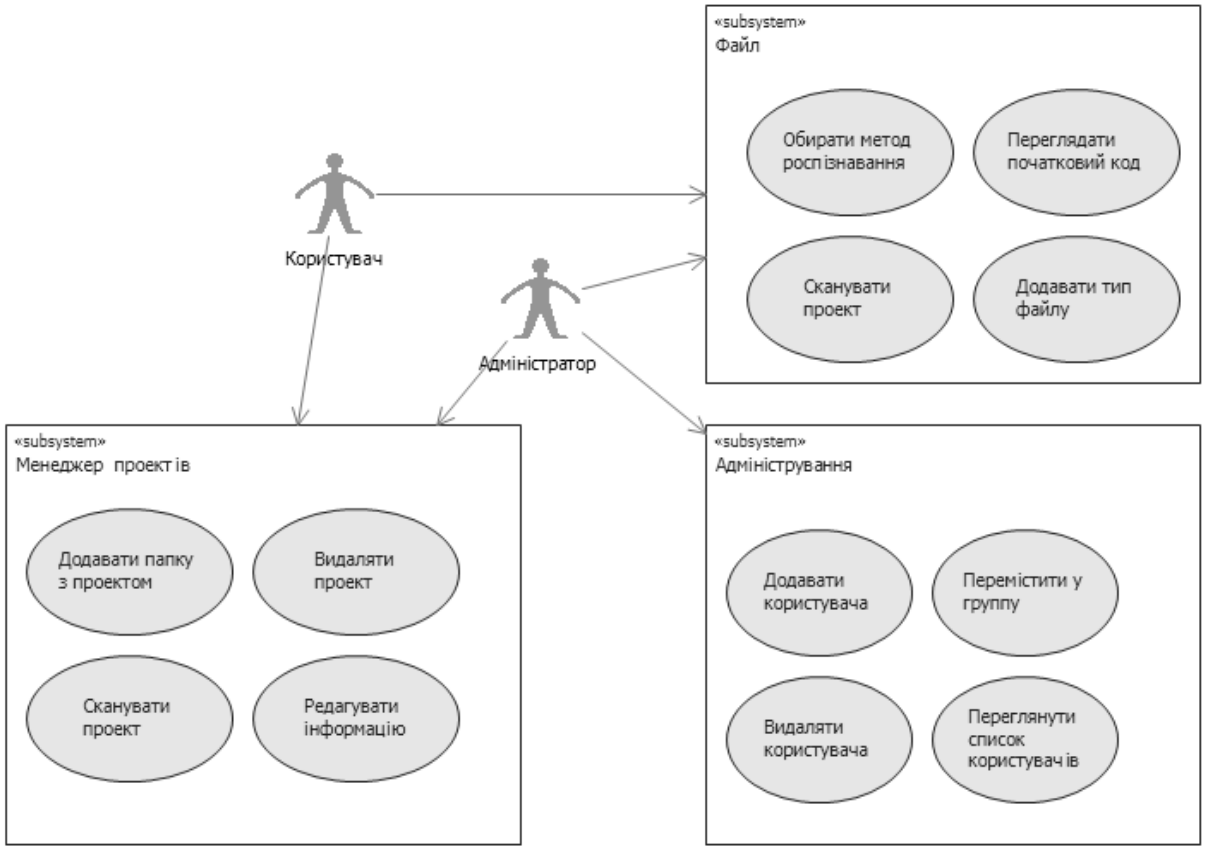

Рис.1. Діаграма прецедентів системи

2.2. Розробка діаграм діяльності

Модель діяльності в *UML* являє собою повідінку системи, як сукупність робіт, котрі можуть виконувати як система, так і актор, при-

чому послідовність робіт може залежати від прийняття певних рішень залежно від умов, що склалися [3,4]. Окрема діяльність (робота) відображається на діаграмі прямокутником із закругленими кутами. Потоки керування роботами визначаються стрілками.

Якщо мова йде про прийняття рішення, то з відповідного прямокутника виходять дві стрілки, на кожній може позначатися текст умови, якій вона відповідає.

Діаграма діяльності (рис. 2) нагадує відомі блок-схеми алгоритмів та програм, зокрема передбачено відображення можливості виконувати паралельно кілька діяльностей і точки синхронізації їх завершення.

Користувач вибирає проект, аналіз якого слід провести. Потім перевіряється умова – чи скінчились файли в проекті. Якщо так, то система переходить до підрахунку кількості плагіату у проекті. В іншому випадку перевіряється чи містить файл код мови програмування.

Якщо містить, то система перевіряє файл, знаходить токени і перетворює початковий код програми на зашифровану послідовність згідно з таблицею токенів.

Далі послідовність розширюється шляхом створення надлишкових комбінацій токенів і утворення k – грамів та, в залежності від обраного методу, система або просто підраховує кількість збігів з іншими файлами, згідно алгоритму Хескела, або формує колекцію міток за алгоритмом відбитків. Потім робиться їх просіювання і тільки після цього порівняння на збіг з іншими файлами у базі даних системи.

Для реалізації знаходження плагіату в системі вибраний алгоритм ідентифікації міток, який є більш потужним за алгоритм Хескела [5].

Якщо файл не містить коду, або має нестандартне для файлів коду розширення, він перевіряється на збіжність з іншими фалами по розширенню, даті останньої зміни, та об’єму.

Всі порушення фіксуються у базі даних.

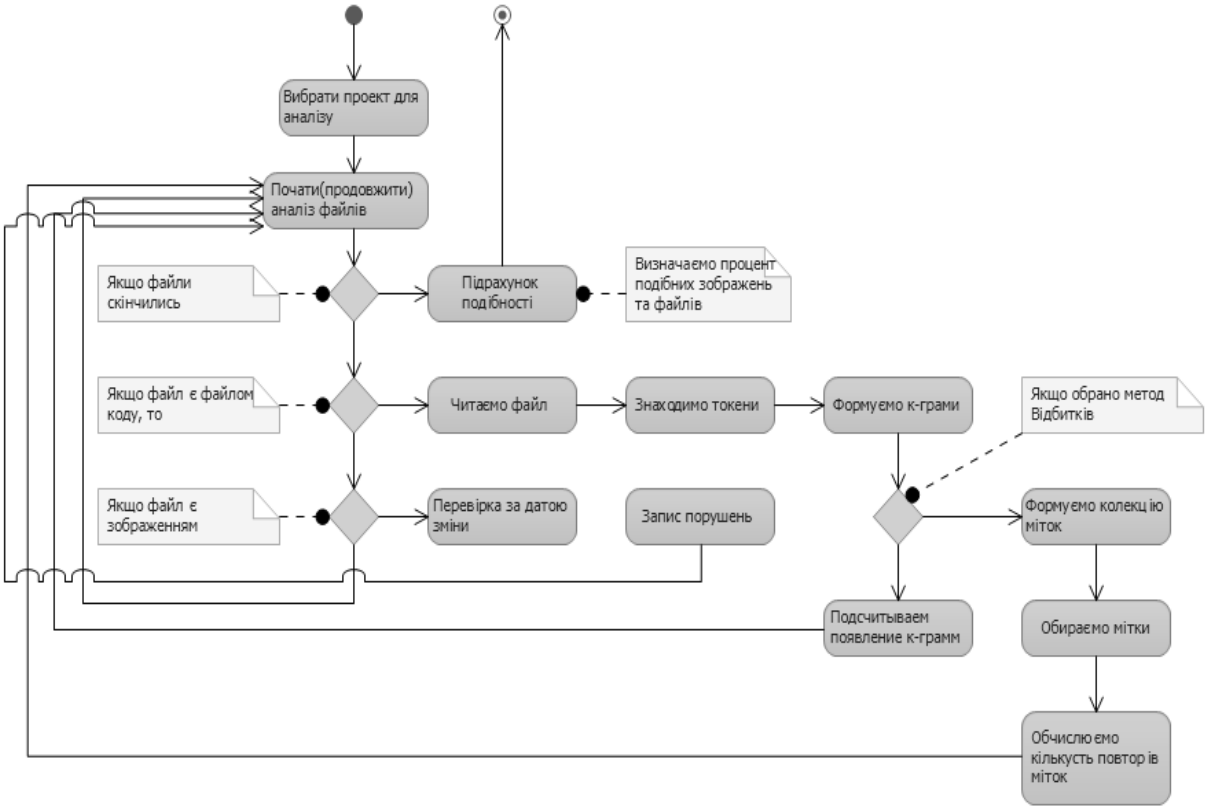

Рис. 2. Діаграма діяльності системи оцінки ідентичності програмного коду

2.3. Розробка моделі класів та об'єктів

На основі діаграм, наведених вище, розроблено моделі класів та об'єктів для системи оцінки ідентичності програмного коду.

Модель містить абстрактний клас *Base*, який наслідується усіма класами. Він містить властивість *Id*, що відповідає унікальному ідентифікатору, *Name* імені та статичну колекцію *ItemList*, що містить усі створені об'єкти даного класу, а також методи *GetItemById()* та *GetItemByName()*, отримання об'єкту з колекції по унікальному ідентифікатору, та по імені відповідно.

Клас *Group* відповідає групі користувачів, для подальшої можливості авторизації користувачів системи.

Клас *Man* – це сам користувач. Крім стандартних властивостей як ім'я, пароль, пошта, він має метод *GetProjectList()* що, дозволяє отримати колекцію проектів, що належить цьому користувачеві.

Клас *Project, Folder* та *File* описують проект, а також теки та файли, що містяться в ньому. Слід зазначити, що файл має властивості, що дозволять відслідковувати дату останньої зміни файлу, об'єм та розширення. Це дозволяє здійснювати більш точний пошук, за цими ознаками файлів, що не мають розширення файлу коду.

Саме клас *FileType* дозволяє відділити пошук по програмному коду, та за ознаками, наведеними вище.

Класи *Token, Kgramm, Mark* потрібні для реалізації алгоритмів знаходження плагіату.

А класи з позначкою *Statistic* потрібні для збереження інформації про пошук в них *k* – грамів, чи міток.

Діаграму класів системи зображено на рисунку 3.

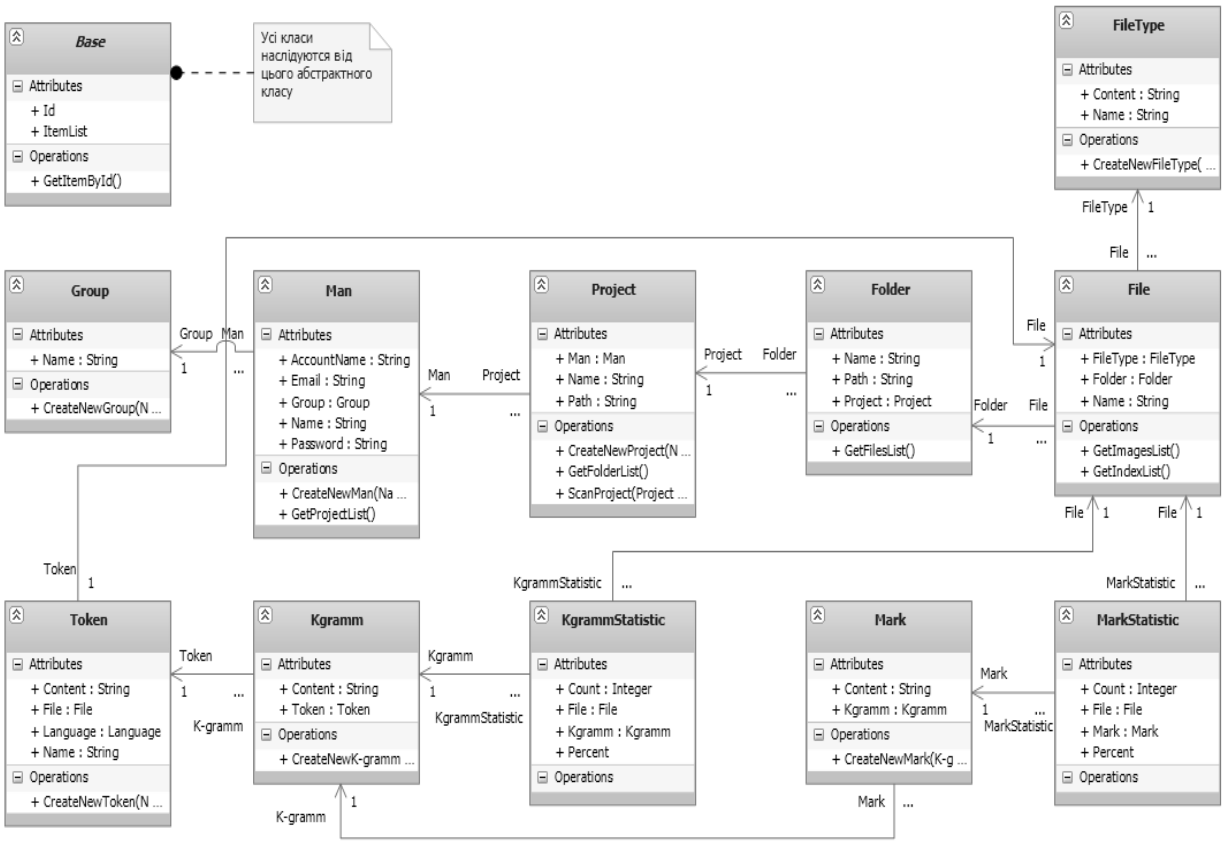

Рис.3. Діаграма класів системи оцінки ідентичності програмного коду

Діаграма послідовності (рис.4) описує етапи створення об'єкту групи.

Користувач системи надсилає запит до головної форми, шляхом натиснення на кнопку «Створити групу». З форми викликається ко-

манда *AddGroupCommand.Execute()* після перевірки можливості її виконання. Потім викликається форма створення нової групи, до якої потрібно ввести дані.

При натисненні на кнопку «Підтвердити» створюється відповідний до введених даних об’єкт класу *Group*. Контроль передається до моделі відображення, що оновлює список груп.

Слід зазначити, що при проектуванні було використано патерн проектування *Model View ViewModel.*

Найважливіший момент *WPF*, що робить *MVVM* дуже зручним шаблоном, є інфраструктурою прив'язки даних.

За рахунок прив'язки властивостей відображення до моделі відображення виходить слабке зв'язування цих компонентів, яке повністю звільняє розробника від необхідності писати в моделі відображення код, безпосередньо його оновлюючий.

Система прив'язки даних підтримує також перевірку введення, яка забезпечує стандартний шлях передачі помилок при перевірки допустимості введеної інформації.

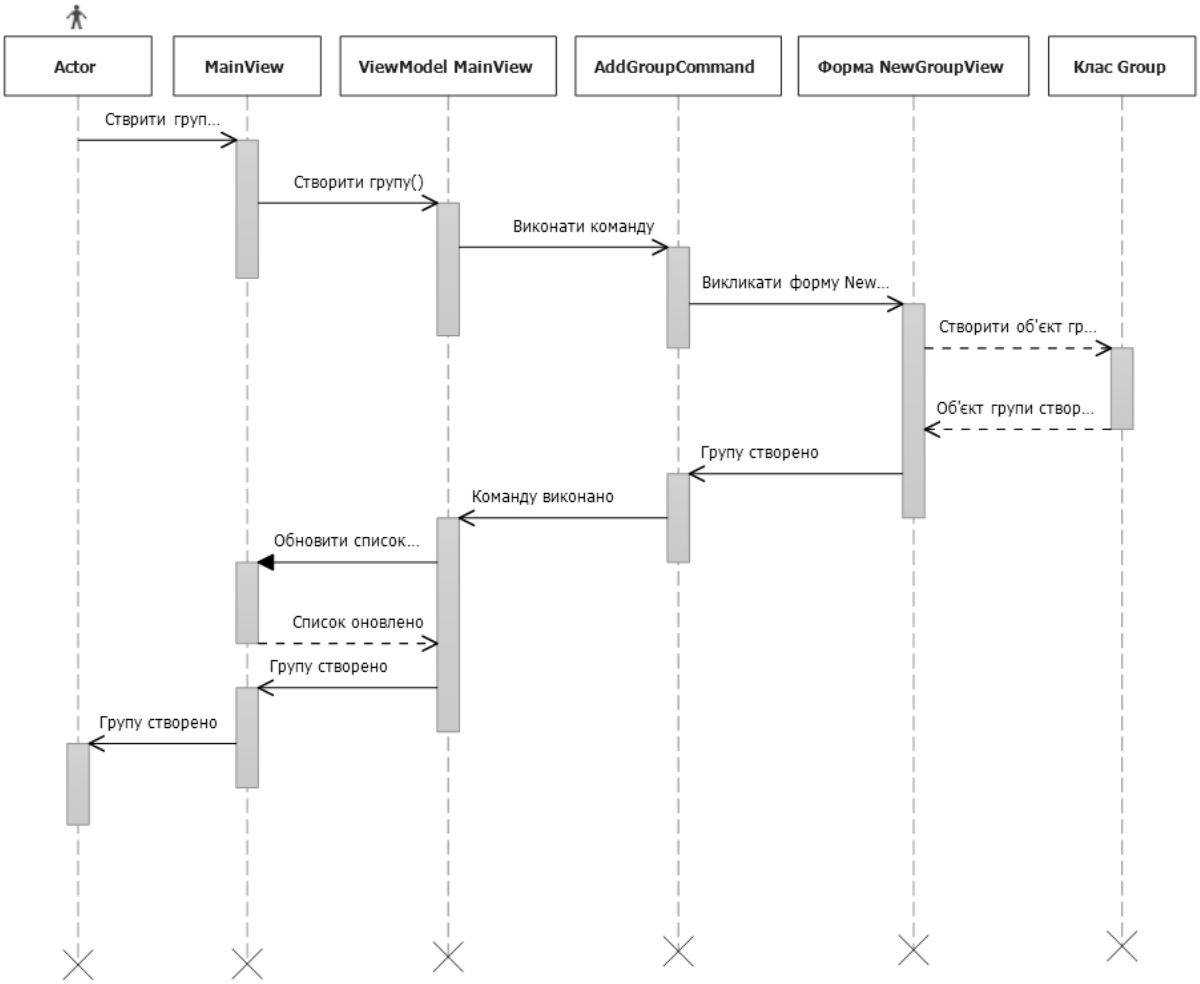

Рис.4. Діаграма послідовності створення класів системи

3. Реалізація моделі системи

Розглянуто поетапну реалізацію моделі запропонованої системи. Проведено налагодження роботи *SQL Server 2008* [6] та визначено загальну структуру проекту.

Наведено етапи розробки програмних модулів адміністрування та проведення аналізу з питань використання і наповнення бази даних.

Структура бази даних для даної розробки має вигляд зображений на рисунку 5.

Структура таблиць в СКБД спроектована згідно з наведеною діаграмою класів та діаграмою об’єктів.

Зв'язок між *Data Model Layer Data* та *Access Layer*, реалізовано кодом *C#.*

Згідно патерну проектування *MVVM* при розробці було створено клас, що реалізує подію зміни відображення. Далі наведено його початковий код.

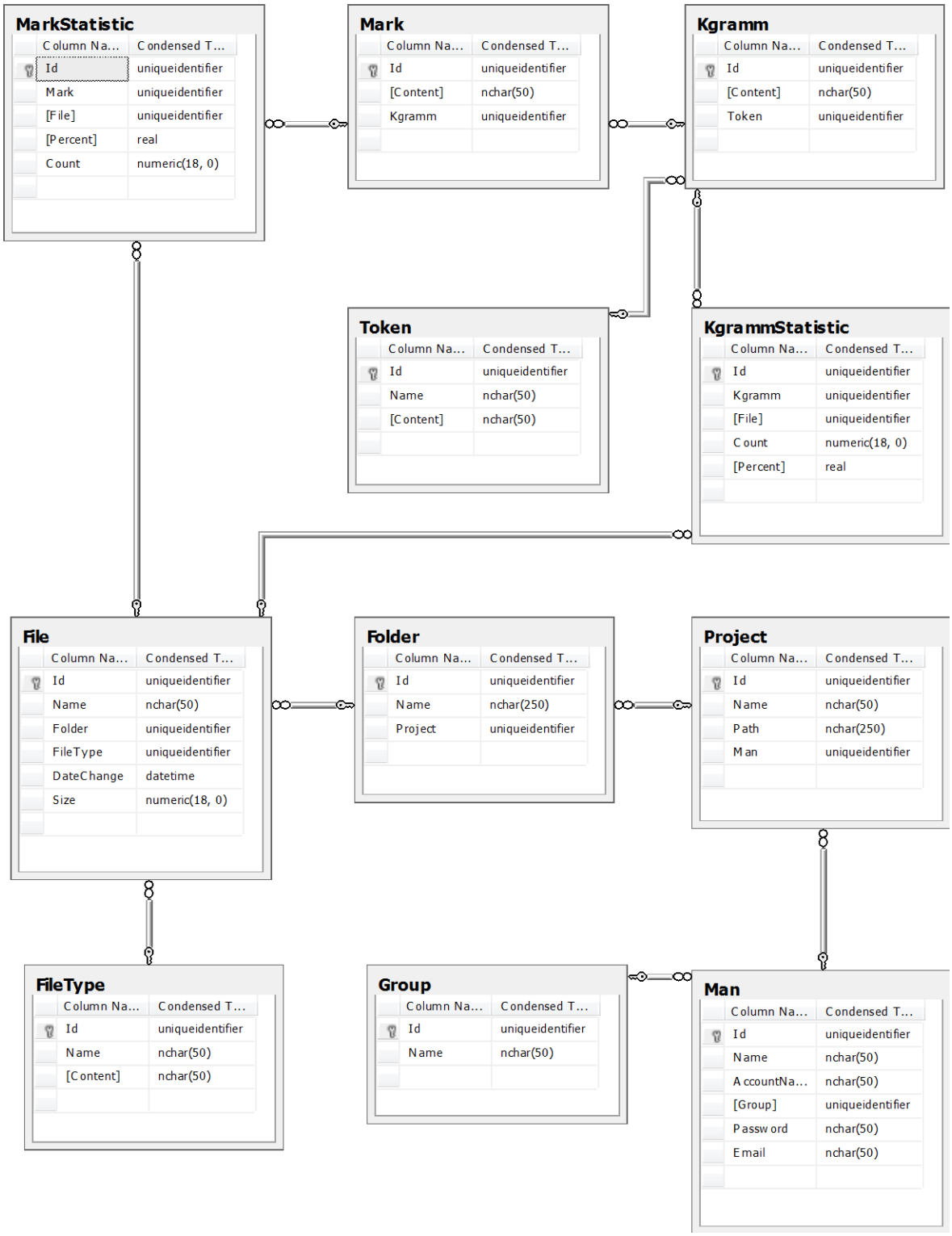

Рис.5. Структура бази даних системи оцінки ідентичності програмного коду

```
public abstract class ViewModelBase : INotify-
PropertyChanged
    {
     #region INotifyPropertyChanged Members
   public event PropertyChangedEventHandler
PropertyChanged;
     #endregion
     #region Protected Methods
     /// <summary>
     /// Fires the PropertyChanged event.
     /// </summary>
   public void NotifyPropertyChanged(String info)
     {
      if (PropertyChanged != null)
       {
        PropertyChanged(this, new Property-
ChangedEventArgs(info));
          }
      }
       #endregion
     }
```

Для розробки візуального інтерфейсу панелі задач рекомендоване використання *Microsoft Ribbon for WPF*, яке дозволить зробити дизайн панелі. Для його використання треба завантажити і встановити додаток.

При використанні в розмітку *XAML* додати:

```
xmlns:ribbon="clr-namespace: Microsoft.Win-
dows.Controls.Ribbon; assembly=Ribbon Controls
Library"
```

А також перетворити вікно на: *ribbon:RibbonWindow*

Згідно з розробленою структурою бази даних для роботи з системою треба створити наступні вкладки на панелі задач:

– профілювання – дозволяє редагувати користувачів системи, створювати і назначати групи, а також редагувати список файлів;

– вкладка аналізу, дозволяє обрати алгоритм, за яким проводиться порівняння, встановити параметри розміру *k* – грама та величину вікна у методі ідентифікації міток.

Перелік команд та їх призначення наведено в таблиці 1.

Таблиця 1. Перелік команд системи

About	Відображення вікна з інформацією про систему
AddGroup	Команда створення групи
AddMan	Створення користувача
AddProject	Додавання проекту
DeleteGroup	Видалення групи
DeleteMan	Видалення користувача
DeleteProject	Видалення проекту
FileScan	Сканування файлу
MakeKgramm	Створення масиву *k* – грамів
ScanProject	Сканування файлу на плагіат
SelectTabGroup	Вибір вкладки “Групи” панелі “Адміністрування”
SelectTabMan	Вибір вкладки “Користувачі” панелі “Адміністру-
SelectTabProject	Вибір вкладки “Проекти” панелі “Адміністрування”
ShowText	Ввімкнути/вимкнути показ тексту файлу

Наслідуючи модель відображення від цього абстрактного класу, отримали можливість оновлювати всі відображення об’єкту лише виконуючи метод *NotifyPropertyChanged*.

Нижче наведено приклад використання описаного методу:

```
public File ViewFile
   {
     get { return _viewfile; }
     set
     {
      _viewfile = value;
      NotifyPropertyChanged("ViewFile");
      if (ShowText == Visibility.Visible)
       {
         NotifyProperty-
Changed("ViewDocument");
          }
       }
     }
```

Також використано метод command binding. Для цього клас, що описує команду унаслідується від інтерфейсу *ICommand* і реалізує такі його властивості:

```
public event EventHandler CanExecuteChanged
    {
      add { CommandManager.RequerySuggested += value; }
     remove { CommandManager.RequerySuggested -= value; }
      }
   public new bool CanExecute(object parameter)
     {
       return (_mViewModel != null);
      }
   public new void Execute(object parameter)
     {
      _mViewModel.About();
      }
```

– *CanExecuteChanged* – необхідний для оповіщення відображення про зміну статусу команди;

– *CanExecute* статус команди, який може, чи може виконуватися у даний момент часу;

– *Execute* – метод, що виконується при виклику команди.

Висновки

В результаті проведеної роботи було розроблено модель системи оцінки ідентичності програмного коду та визначені шляхи її реалізації.

Проектування проводилось за об'єктно-орієнтованою парадигмою.

Усі необхідні діаграми проектувалися за допомогою *CASE*-засобу *IBM Rational Rose* [7]. Було розроблено гнучку модель системи з огляду на існуючі ризики.

З метою покращення подальшої підтримки та розвитку системи, вихідний код програми написано в рамках сучасних технологій з дотриманням стандартів написання коду, які рекомендовані Microsoft та є максимально декларативним.

Система виконує поставлені перед нею задачі, а саме: реєстрацію користувачів, створення груп користувачів; імпорт проекту з усією внутрішньою структурою; аналіз на плагіат (збіг структури каталогів, ідентичні зображення, файли з однаковою датою змінення).

Вона відповідає усім вимогам та реалізує поставлені задачі.

При проведення подальших досліджень з питань розробки системи планується наступне:

– реалізувати загальні модулі та інтерфейс системи;

– доповнити систему модулями пошуку інших алгоритмів, задля розширення переліку типів файлів з якими вона може працювати;

– для розширення аудиторії з використання системи розробити веб – інтерфейс користувача.

СПИСОК ЛІТЕРАТУРИ

1. *Stepanova E.B., Krivtsov V.E.* Process modeling in Educational IT-Projects. CSIT'2008: Proceedings of the 10-rd International Workshop on Computer Science and Information Technologies (Antalya, Turkey, September 15-17, 2008). – Ufa: Ufa State Aviation Technical University, 2008. – v. 1. – pp. 227-230.

2. *Faidhi J.A., Robinson S.K.* An Empirical Approach for Detecting Program Similarity with in a University Programming Environment. – Computers and Education,1987. – № 11(1) . – pp. 11-19.

3. *Рамбо Дж., Блаха М.* UML 2.0. Объектно-ориентированное моделирование и разработка. 2-е изд. – СПб.: Питер, 2007. – 544 с.: ил.

4. *Мюллер Р.* Базы данных и UML, проектирование. – М.: Лори, 2002. – 420 с. – ISBN: 5-85582-168-4

5. *Baker B.S.* On Finding Duplication and Near-Duplication in Large Software Systems. In Proceedings of the second IEEE Working Conference on Reverse Engineering (WCRE), July 1995. – pp. 86–95.

6. Опис SQL Server 2008 R2 [Електронний ресурс]: Режим доступу: http://www.microsoft.com/sqlserver/en/us/default.aspx. – Назва з екрана.

7. *Трофимов С.А.* CASE-технологии: практическая работа в Rational Rose. Изд. 2-е. – М.: Бином-Пресс, 2002 г. – 288 с.: ил.

G.G. Kirichek, A.A. Kirichek. **Model of program code identity estimation system.**

Г.Г. Киричек, А.А. Киричек. **Модель системы оценки идентичности программного кода.**

Предложена модель системы оценки идентичности программного кода для использования при изучении дисциплин по программированию. Исследование посвящено решению актуальной задачи повышения эффективности изучения и практического применения языков программирования в процессе обучения. Система обеспечивает анализ и обработку результатов разработки программных проектов, выполненных студентами и определение случаев повторного использования кода.