Проведен анализ методов оценки усталости металла на основе прочностных, деформационных и физических критериев. Установлена целесообразность использования средств контроля, диагностические параметры которых изменяются монотонно, в соответствии с общим процессом исчерпания ресурса изделия.

Ключевые слова: метод контроля, диагностический параметр, остаточная долговечность

The analysis of assessment methods based on fatigue strength, deformation and physical criteria. The expediency of the use of controls, diagnostic parameters which change monotonically, in accordance with the general process of exhaustion of the resource products.

Keywords: method of monitoring, diagnostic parameters, the residual life.

УДК 004.43+004.9

А. А. КИРИЧЕК, магистр, НТУУ “КПИ”, Киев;
А. А. АМОНС, канд. техн. наук, доц., НТУУ “КПИ”, Киев;
Г. Г. КИРИЧЕК, канд. техн. наук, доц., НТУУ “КПИ”, Киев

АЛГОРИТМ ФИЛЬТРАЦИИ ДЛЯ СИСТЕМЫ ОПРЕДЕЛЕНИЯ ПЛАГИАТА В ПРОГРАММНОМ КОДЕ

Предложен новый алгоритм фильтрации, как часть системы определения плагиата в программном коде. Исследование посвящено решению задачи отсечки файлов шаблона и фрагментов программного кода проекта до применения основных алгоритмов оценки подобия.

Ключевые слова: плагиат, программирование, токенизация, синтаксический анализ, фильтрация.

Введение. Задачей системы оценки идентичности программного кода, является автоматическое обнаружение (по заданным критериям) того, была ли использована в программе чужая идея. На практике определенным образом задаются функция близости и порог, по которым можно определить насколько вероятно, что определенная часть кода была заимствована [1].

В соответствии с [2], принято выделять следующие подходы к оценке близости программ: атрибутно-подсчетный, структурный и комбинированный, сочетающий в себе первых два.

Смысл атрибутных методов в численном выражении некоторых признаков (атрибутов) программы и сравнении полученных чисел для разных программ. Программы с близкими численными характеристиками атрибутов потенциально похожи. Две программы считаются похожими, если соответствующие числа из их наборов совпадают или близки [3], поэтому оценка близости программ сводится к сравнению чисел или векторов, получаемых путем простого анализа непосредственно исходного кода. Недостатком атрибутных техник является то, что несвязанные между собой параметры программы плохо описывают ее в целом. При таком подходе разные программы получают близкие характеристики [2].

Структурные методы исследуют свойства программы не изолированно, а как бы в контексте, устанавливают взаимосвязь различных характеристик и их совместное поведение. Для выделения нужных зависимостей, программа переводится в более компактное представление (токенизация исходного кода). Классический пример структурного подхода - построение дерева программы с последующим сравнением деревьев для разных программ. Недостатком структурных методов является их сложность и вычислительная трудоемкость. Кроме того, они обычно опираются на синтаксис конкретного языка программирования [4]. Адаптация метода для другого языка требует значительных усилий. Сложность реализации алгоритмов, использующих структурные методы, является платой за точность этих алгоритмов [5].

Комбинированный подход целесообразно использовать для поиска плагиата в большой базе программ [6]. Для этого на первом этапе с помощью одного из атрибутных

методов можно отсеивать заранее непохожие программы. На втором этапе выполняется более детальное сравнение оставшихся программ каким-либо структурным методом [7]. Таким образом, за счет предварительного несложного анализа сокращается количество парных сравнений при поиске плагиата, а, следовательно, растет эффективность.

Детекторы плагиата, основанные на этих методах [2] применяются для текстов исходного кода и не проводят анализ на наличие участков кода и файлов проекта, которые являются автоматически генерируемыми при создании проекта на основе шаблона. Это вносит ошибку в результирующий процент сходства проектов.

Постановка задачи. Все текущие алгоритмы ориентированы на поиск плагиата в коде отдельных файлов. Зачастую проекты, разрабатываемые студентами, состоят из множества файлов программного кода (один класс – одни файл), чаще всего использующих стандартные шаблоны [8].

Актуальность и целесообразность данного исследования заключается в необходимости уточнения текущих алгоритмов определения повторного использования программного кода, за счет применения к разработанному проекту алгоритма фильтрации, запускаемого на этапе импортирования в систему. Благодаря реализации такого алгоритма появится инструмент, позволяющий импортировать в систему проект целиком и получать более высокую точность оценки наличия в нем плагиата без учета кода, который генериуется средой разработки и является стандартным.

В данной работе рассматривается решение поставленной задачи для проектов, написанных на языке программирования C# [5]. Большая часть кода простейших программ на C# имеют практически идентичную реализацию в связи с тем, что они созданы на шаблонах проектов предлагаемых средой разработки.

Для выделения уникальной составляющей, выполненного студентом задания, следует применить отсеивание. Удаление из исследуемого проекта файлов программного кода шаблона позволит оценить изменения, внесенные конкретным разработчиком. Без такого отсеивания очень часто получаются работы очень сильно похожие (до 60-80%) за счет того, что автогенерированный код в обоих проектах идентичен и имеет большой процент по отношению коду, написанному вручную.

Алгоритм фильтрации шаблонного кода. Реализация предложенного алгоритма фильтрации состоит из двух частей. Первая исключает определенные файлы из проекта, являющиеся частью предопределенных шаблонов, на основе которых он был создан.

Вторая полностью исключает или существенно уменьшает вклад в общий процент подобия участков кода, которые соответствуют определенным шаблонам среды Visual Studio, либо реализации паттернов проектирования[1]. Комплексное применение данных методов позволит повысить точность последующей оценки подобия программного кода.

Алгоритм обработки проекта, перед проведением его анализа, представлен на рис.1.

Его суть в проведении фильтрации шаблонных файлов проекта. Основные этапы для всех файлов кода, включают в себя выполнение последовательности операций, состоящей из: удаления комментариев, автоформатирования, токенизации, а также поиска и исключения

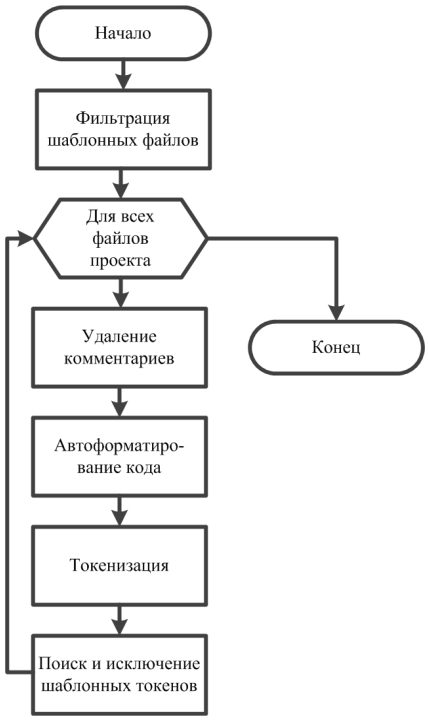

Рис.1 - Алгоритм обработки проекта перед анализом на плагиат

шаблонных токенов.

На первом этапе отсекаются файлы, которые содержат только автогенерированный код. После фильтрации файлы одного проекта связываются в один большой блок файлов для дальнейшей обработки. Далее для каждого файла в блоке выполняется последовательно удаление комментариев и автоформатирование кода к единому стилю. Это позволяет в дальнейшем сократить количество несущественных токенов. После форматирования кода выполняется его токенизация. Для полученных токенов на следующем шаге на основе шаблонов токенов и формальных правил выполняется поиск шаблонных токенов в наборе и исключение их.

На выходе данного алгоритма мы получаем токенизированное представление проекта из которого исключены стандартные и шаблонные токены, что позволяет выделить из массы программного кода существенные для последующей оценки части.

Реализация алгоритма. В процессе получения информации о заимствовании определенных участков программного кода в разработанных проектах, имеется возможность произвести их оценку и представить отчет об использовании идентичных файлов ресурсов, либо отдельно взятых модулей приложения. Это даст возможность преподавателю более точно оценить уникальный вклад студента в решении, поставленной перед ним задачи [7].

Для исключения файлов шаблонных проектов, используется метод поиска в каталоге проекта предопределенных файлов, входящих в состав каждого шаблона. Предустановленные в среду разработки шаблоны можно дополнять пользовательскими, которые становятся доступны при создании нового проекта.

Для примера можем рассмотреть шаблон WPF MVVM project template. Структура данного шаблона предоставлена на рис. 2.

Проект включает в себя список файлов, которые должны быть исключены из проверки, но только при соблюдении условия, что в них не вносятся изменения в процессе разработки самого программного проекта.

Данный метод можно применять к любому типу шаблонов. При необходимости можно проанализировав информацию о вычитании определенных шаблонных файлов из проекта предоставить отчет о проценте использования стандартных шаблонов.

Шаблоны импортируются в систему путем добавления проекта созданного на их основе без внесения каких бы то ни было изменений. Система хранит информацию о файлах в виде названий и контрольных сумм.

После того, как не подлежащие изменению файлы исключены из проекта, необходимо проанализировать все оставшиеся файлы проекта на наличие фрагментов кода, состоящих из стандартных шаблонов.

Первым этапом подготовки файла к исследованию является проведение его очистки от комментариев. Наличие подобия данных участков не должно оказывать влияние на общее оценивание разработанного проекта.

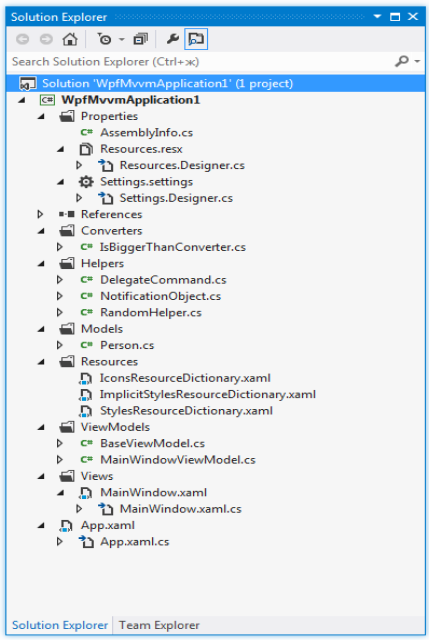

Рис.2 - Структура шаблона для создания проекта

В данном случае исключению подлежат блочные и строковые комментарии. Ниже приведен код программы, выполняющий очистку комментариев с учетом исключений:

public string DeleteComments(string input)

```
{
        var blockComments = @"/\*(.*?)\*/";
    var lineComments = @"//(.*?)\r?\n";
    var strings = @"""((\\[^\n]|[^""\n])*)""";
    var verbatimStrings = @"@(""[^""]*"")+";
    return Regex.Replace(input, blockComments + "|" + lineComments + "|" + strings + "|"
+ verbatimStrings,
        me =>
        {
          if (me.Value.StartsWith("/*") || me.Value.StartsWith("//")) return
me.Value.StartsWith("//") ? Environment.NewLine : "";
          // Keep the literal strings
          return me.Value;
        },
        RegexOptions.Singleline);
}
```

Обязательным является выполнение этапов автоформатирования кода. Данная процедура необходима для расстановки интервалов между операторами. В результате ее применения исключаются ошибки при проведении процесса токенизации и, соответственно, уменьшается сложность реализации самого процесса.

Выполнить процесс автоформатирвания можно, воспользовавшись средствами Microsoft Visual Studio. Для этого используется библиотека EnvDTE.dll. Это COM библиотека, она содержит объекты и элементы для автоматизации ядра Visual Studio. Ниже приведен код, позволяющий выполнить команды автоформатирования для нужного файла:

```
addedItem = project.ProjectItems.AddFromTemplate(file.FullName, fileName);
addedItem.Open(Constants.vsViewKindCode);
addedItem.Document.Activate();
addedItem.Document.DTE.ExecuteCommand("Edit.FormatDocument");
addedItem.SaveAs(file.FullName);
```

Далее необходимо удалить из файлов шаблонные участки кода. Этого можно достичь, исключив токенизированные представления этих фрагментов. Рассмотрим шаблон проекта Windows Presentation Foundation:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;
using System.Windows.Controls;
namespace WpfApplication2
{
  public partial class MainWindow : Window
  {
    public MainWindow()
    {
      InitializeComponent();
    }
  }
}
```

Далее приведем токенизированное представление этого фрагмента программы.

Допустим, что определению области имен соответствует токен *a*, модификаторам public и partial токены *b* и *c*, а ключевому слову class токен *e*, тогда вложенный токен будет иметь вид:

$$a\{bcd\{b\}\}\},$$

который при указании k-грамма, будет приравниваться одному символу. Например, если k=3, то он будет иметь вид (рис. 3), где $a\{bcd\{b\}\}$ – первый символ, k – второй, m – третий.

Этот фрагмент кода повторяется для многих файлов в проекте при условии, что они используют паттерн проектирования Model-View-ViewModel.

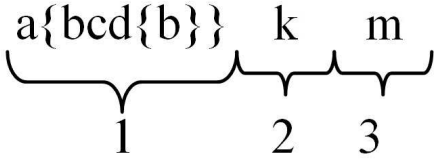

Рис.3 - Вложенный токен

Данный паттерн проектирования используется для разделения модели и её представления. Это необходимо для осуществления их независимости друг от друга в процессе внесения в них изменений. Например, когда программист задает или корректирует логику работы с данными, а дизайнер, соответственно, вносит изменения в работу с пользовательским интерфейсом.

Таким образом, эти повторяемые фрагменты кода могут повысить процент подобия двух абсолютно разных программных проектов. Из этого следует, что их необходимо тоже исключить из процесса проведения проверки на наличие плагиата в программном коде.

Если же подобные файлы содержат логику обработки событий, то они должны принимать участие в процессе проведения оценки уровня подобия. Пример сигнатуры метода приведен ниже:

```
private void btnSave_Click(object sender, RoutedEventArgs e)
{
}
```

При этом содержание файлов с расширением “.xaml.cs” должно быть проанализировано на наличие единственного составного токена.

Автоматически генерируемые файлы с расширением “.Designer.cs” также должны быть исключены из процесса проведения анализа программного кода, если их содержимое соответствует уже предопределенному токену. Пример кода такого файла приведен ниже:

```
namespace WpfApplication2.Properties
{
   [global:: System.Runtime.CompilerServices. CompilerGeneratedAttribute()]
   [global::System.CodeDom.Compiler.GeneratedCodeAttribute(
"Microsoft.VisualStudio.Editors.SettingsDesigner .SettingsSingleFileGenerator",
"11.0.0.0")]
   internal sealed partial class Settings : global::
System.Configuration.ApplicationSettingsBase
   {
      private static Settings defaultInstance = ((Settings)(global::System.
Configuration.ApplicationSettingsBase.Synchronized(new Settings())));
      public static Settings Default
      {
         get
         {
            return defaultInstance;
         }
      }
   }
}
```

На рис. 4 изображена структурная схема, выполнения процесса обработки файлов проекта. Импортируя проект мы выделяем те из файлов, которые не соответствуют шаблонным. Правила фильтрации файлов определяют к каким форматам файлов будет применен алгоритм фильтрации.

Рис.4 – Структурная схема процеса обработки файлов

На данном этапе отсеиваются файлы ресурсов, изображений и т. д. если они являются частью одного из известных шаблонных проектов. Результирующий перечень файлов разделяется на две группы. Одна включает файлы ресурсов, вторая файлы программного кода для дальнейшей обработки.

Файлы программного кода подвергаются процедуре удаления комментариев и автоформатированию. Далее производиться преобразование программного кода в токенизированное представление и производится поиск шаблонных токенов, в результате чего получается множество токенизированых представлений файлов проекта.

Полученный перечень файлов ресурсов и множество токенизированых представлений файлов предоставляется системе оценки плагиата в программном коде для дальнейшего анализа.

Выводы. В результате проведенного исследования предложен подход к предварительной обработке програмного проекта перед оценкой подобности програмных проектов. Решена задача отсечки файлов шаблона и фрагментов программного кода, в разработанных программных проектах, до применения основных алгоритмов оценки подобия.

Полученный алгоритм обеспечивает исключение из проекта неизменяемых фрагментов кода, являющихся шаблонами, в результате чего увеличена точность определения наличия плагиата в программном коде. Также при подготовке файлов проекта к последующему анализу на плагиат в данный алгоритм включены, пошагово, процессы автоформатирования и удаления из кода комментариев. Данный алгоритм фильтрации реализовывается как часть системы определения плагиата в программном коде, разработанных студентами проектов. Система разрабатывается на платформе .Net Framework 4.0 для проектов на C#. В системе собраны правила фильтрации кода для проектов Visual Studio 2012. Разработка системы ведется на кафедре АУТС НТУУ «КПИ».

Список литературы: **1**. *Киричек Г. Г.* Модель системи оцінки ідентичності програмного коду [Текст] / *Г. Г. Киричек, О. О. Киричек* // Науковий вісник Чернівецького національного університету. Комп'ютерні системи та компоненти. – Т. 2. – Вип.3. – 2011. – С. 14-21. **2.** Обзор автоматических детекторов плагиата в программах [Электронный ресурс].- Режим доступа: http://logic.pdmi.ras.ru/~yura/ detector/survey.pdf. - Загл. с экрана. **3.** *Dr Richard Li-Hua*. From technology transfer to knowledge transfer – a study of international joint venture projects in China [Заглавие с экрана] / *Dr Richard Li-Hua*.- Режим доступа: http://knowledgemanagement.uk.net/ resources/Richard Lihua paper.pdf **4.** *O'Sullivan Bryan*. Distributed revision control with Mercurial [Заглавие с экрана] / Bryan O'Sullivan. – 2007. – 211p.: Режим доступу: http://hgbook.red-bean.com/hgbook.html. **5.** *Рамбо Дж*. UML 2.0. Объектно-ориентированное моделирование и разработка [Текст] / *Дж. Рамбо, М. Блаха*. – СПб.: Питер, 2007. – 544 с.: ил. **6.** *Stepanova E. B.* Process modeling in Educational IT-Projects [Текст] / *E. B. Stepanova, V. E. Krivtsov* // 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. **7** *Киричек Г. Г.* Модель оцінки плагіату програмного

коду на основі системи контролю версій [Текст] / *Г. Г. Киричек, О. О. Киричек* // Східно-європейський журнал передових технологій.– Харків: Технологічний центр, 2012.– №2/2(56).– С.25-32. **8**. *Макаров В. В*. Идентификация дублирования и плагиата в исходном тексте прикладных программ [Электронный ресурс] / *В. В. Макаров*.- Режим доступа: http://lab18.ipu.ru/projects/conf2006/1/15.htm.- Загл. с экрана.

Надійшла до редколегії 20.03.2013

УДК 004.43+004.9

Алгоритм фильтрации для системы определения плагиата в программном коде/ А. А. Киричек, А. А. Амонс, Г. Г. Киричек // Вісник НТУ «ХПІ». Серія: Нові рішення в сучасних технологіях. – Х: НТУ «ХПІ», – 2013. - № 1 (977). – С. 76-82. – Бібліогр.: 8 назв.

Запропоновано новий алгоритм фільтрації, як частина системи визначення плагіату в програмному коді. Дослідження присвячено вирішенню задачі відсічення файлів шаблону і фрагментів програмного коду проекту до застосування основних алгоритмів оцінки подібності.

Ключові слова: плагіат, програмування, токенізація, синтаксичний аналіз, фільтрація

A new filtering algorithm, as part of the system definition plagiarism in the program code. Research is devoted to the task of clipping template files and program code fragments from the project before applying the basic algorithms for the assessment of similarity.

Keywords: plagiarism, programming, tokenization, parsing, filtering.

УДК 621.391

В. В. КОРЧИНСКИЙ, канд. техн. наук, доц., ОНАС им. А.С. Попова, Одесса

КОНФИДЕНЦИАЛЬНАЯ СИСТЕМА СВЯЗИ НА ОСНОВЕ ПСЕВДОСЛУЧАЙНОЙ ПЕРЕСТРОЙКИ РАБОЧЕЙ ЧАСТОТЫ И ТАЙМЕРНЫХ СИГНАЛОВ

Для конфиденциальной системы связи многопользовательского доступа предложен метод формирования группового сигнала на основе псевдослучайной перестройки рабочей частоты и таймерных сигналов.

Ключевые слова: таймерный сигнал, псевдослучайная перестройка рабочей частоты.

Введение. Одним из возможных направлений повышения скрытности передачи в конфиденциальных системах связи является использование методов кодирования и передачи данных на основе расширения спектра информационного сигнала с помощью псевдослучайной перестройки рабочей частоты (ППРЧ) [1].

Известно [1], что метод ППРЧ позволяет решать задачи по обеспечению энергетической скрытности передачи сигнальных конструкций на первом уровне эталонной модели OSI. Немаловажным показателем по обеспечению конфиденциальности передачи является структурная скрытность используемых сигнальных конструкций. Этот показатель должен противостоять мерам, которые направлены на распознавание формы сигнальных конструкций, если станцией НСД решена проблема обнаружения факта передачи и перехвата сообщений. В [2-5] была дана оценка структурной скрытности таймерных сигнальных конструкций (ТСК), что позволило сделать вывод о целесообразности их применения в конфиденциальных системах связи. В [6] показана возможность совместного применения метода передачи ППРЧ и ТСК для задачи повышения скрытности передачи сигнальных конструкций.

Дальнейшее развитие метода [6] определяется актуальностью повышения пропускной способности каналов связи при передаче конфиденциальной связи.

Целью работы. Целью работы является разработка метода формирования группового сигнала на основе совместного использования ППРЧ и ТСК при построении конфиденциальной системы связи многопользовательского доступа.

Метод построения группового сигнала. В системе ППРЧ заложена технология преднамеренного расширения спектра информационного сигнала при его передаче, в