Coding Standard Violation Detection by Pattern Analysis

B.V Tishantha Dilruk

158756G

Supervised by Mr. Chaman Wijesiriwardana

Master of Science in Information Technology

University of Moratuwa, Sri Lanka

February 2019

Declaration

I hereby declare that this is my own work and has not been submitted in any form for another degree or diploma at any university or any other institution of tertiary education. Information derived from the published or unpublished work of others has been acknowledged duly in the text and a list of references is given as per the standard.

Name of Student	Signature of Student
B.V. Tishantha Dilruk	………………………
	Date: ………………..

Supervised by

Name of Supervisor	Signature of Supervisor
Mr. Chaman Wijesiriwardana	……………………….
	Date: ………………...

Acknowledgment

Firstly, I want to thank my supervisor Mr. Chaman Wijesiriwardana Senior Lecturer, Faculty of Information Technology, the University of Moratuwa for all the shared wisdom and guidance during the process of making this thesis. Secondly, I would be grateful to Prof. Asoka Karunananda who has continuously guided me for conducting comprehensive research work.

Thirdly I would like to thank all my batch mates specially Nadun and Lahiru who gave me some helpful technical support and also to Dananjaya, Asanga, Kalindu, Kaushalya, and Dilan for their support to the evaluation of this project.

I, of course, owe a special thank you to my family and my girlfriend Hemalie and her family for the support and trust always kept on me throughout the task. Without you, it is impossible to complete this research.

Abstract

Today we live in the era of Information Technology. The success of any other industry is linked with the way how they use Information Technology to handle their operations. In order to fulfill that requirement, presently there are various kinds of software have been developed. Developing software is not that much of an easy task since it has a development lifecycle to build successful software. However, there are some critical issues that we can identify when developing a software project. Software complexity, maintainability, and enhancement are the major issues which we can highlight in our literature review section. Poor cording standard drive makes the most of the software projects complex and extremely difficult to enhance and maintain.

In this thesis we have proposed the coding standard violation detection mechanism by pattern analyzing. With this approach, a standard coding guideline is kept through an online reference using pattern analyzes mechanism. This tool helps the developer to do the developments through a standard guideline. It will also prevent violations done by the programmer. In this tool, it would provide a facility to add a coding standard through the online reference. Then the proposed tool will take it as the model to follow the standard. Whenever the developer violates the standard the error will be shown.

Table of Content

List of Figures

List of Tables

Chapter 1

1. Introduction

1.1 Prolegomena

This chapter we will present an introduction about this thesis by detailing the project background and motivation which lead to start this research. Also, this chapter presents a brief description about the proposed solution as well as identifying the aims and objectives.

1.2 Background and Motivation

Nowadays information system software becomes a key factor to success in any other field in the business industry. However, developing software is a large process and that has a standard called software development life cycle (SDLC). When comes to SDLC, Software development methodologies should be addressed [1]. There are several software development methodologies are available. But a few of them are considered as major software development methodologies. SDLC for particular software is beginning with those methodologies. Selecting a suitable methodology will be depending on the software project. Even the correct methodology is chosen, unstructured development will increase complexity of the project and also increases the time duration to complete the project.[2]
According to the statistics as of 2015, [Figure 1] it can be identified few major reasons for a project failure.[3] Poor documentation or requirement changes, lack of resources, organization or management problems, insufficient time allocation for testing, developers change, delivery time, time constraint and pre-mature software release and Immature development tools and application platforms are the main reasons that can be highlighted. As we can see in the Figure 1, there are 48% of software project failures cased due to requirements changes and poor documentation. Therefore, it is very important to study about the requirements changes and documentation in a software project. Since the software requirement specification (SRS) is a complete description about the project and functions, some projects have become unsuccessful by failing to address the user requirements. In many cases, at the first release the system works properly, but failed to apply an enhancement or a modification.

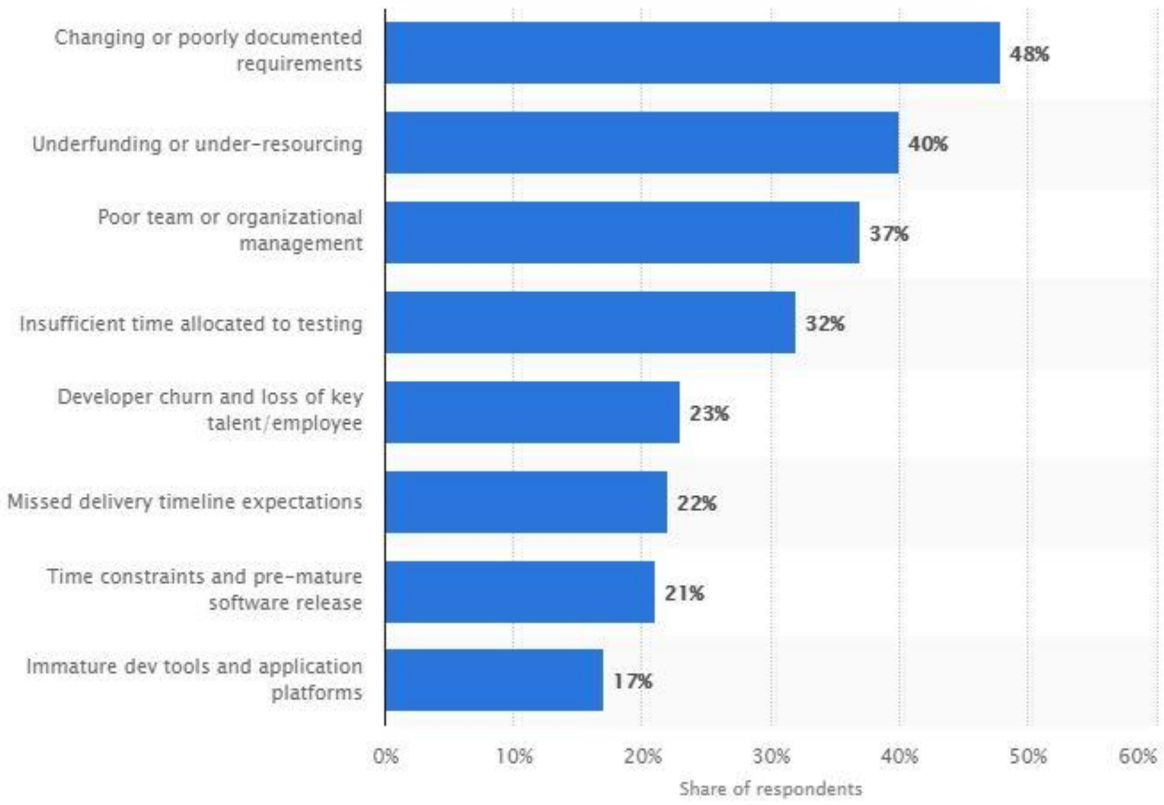


Figure 1- Leading reasons for software project failure according to developers worldwide, as of 2015

To conclude, as we live in the era of Information Technology, there are so many things to consider when developing a software project. After developing most of the software, it is required to maintain continuesly. Especially Enterprise Resource Planning (ERP) and financial applications need to be maintained properly and there can be critical enhancements depending on the industry type and business requirements.

1.3 Problem in Brief

Requirements changing and modification requests are frequent requests in a large software solution. When developing a software solution, developers can follow their own standard to build the codes. Even the solution is working properly, there can be arose many bugs and difficulties in future developments.

1.4 Aim and Objectives

The aim of this research is to implement and maintain a software project according to a set of coding standard captured from the online guideline. In order to achieve that goal, we propose a coding standard violation detection tool using pattern analysis technique to embed with development IDE. Apart from that, the following objectives can be highlighted.

1. Study previous works and identifying gaps
2. Develop a hypothesis
3. Design a tool to detect coding standard violations
4. Implement a tool
5. Evaluate a tool

1.5 Proposed Solution

We proposed a tool to do the software cording through an online guideline, which will prevent the violations of coding standard. This solution will be developed by pattern analysis mechanism which is used in many comparison applications.

1.6 Summary

In this chapter, we describe the over role idea about the document. Next chapter will percent the challenges in software development life cycle and a brief discussion about the background of software development methodologies with the important of coding standard. Also, it will present a detail study about previous researches and exiting tools.

2 Developments and Challenges in Software Development Lifecycle

2.1 Introduction

This section summarizes previous journals related to our research problem and identifies research space we try to address. Firstly, we discuss the evaluation of the coding standard and the importance of the cording standard in future development. Secondly, we review existing problems in cording standard violation in the software industry and discussing their common effects. Thirdly, we review existing empirical research on defects discovered in cording standard violation and find contradicting evidence related to our research problem.

2.2 History of cording standard violation and their effects

Research in software development back dated to mid-1960s[4]. The software development frame works offer facilities to build a software solution by going through the software development life cycle (SDLC) [5]. The process in the SDLC varies across industries and organizations. But standard such as ISO/IEC/IEEE 12207:2017[6] represent processes and provide a mode for the development, acquisition, and configuration of software systems [7]. Implementation and Maintaining are very important parts of SDLC which include the development of the software, modification of existing system and enhancement of the software.

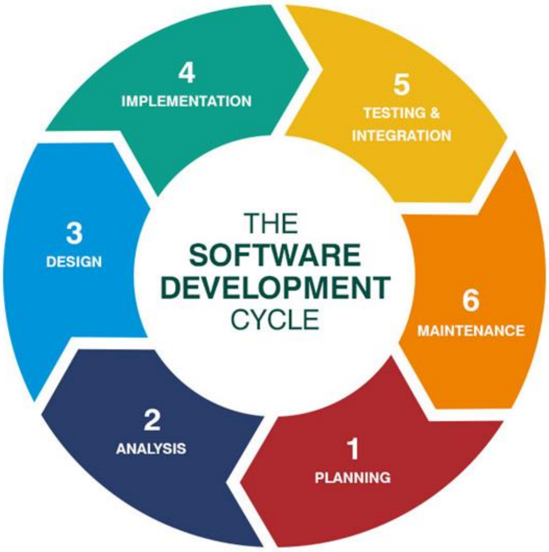


Figure 2-Software Development Life Cycle

When considering a large industrial software like Enterprise Resource Planning (ERP) and financial applications, system modifications and changes are a common thing due to the Industry and business domain[8]. However, according to the statistics when developing those modifications, it will cause for the most of the software failures[3]. And as the major reason for this issue is software complexity, due to coding standard violation can be identified. There are some studies available which addresses similar problem as discussed in our research.

Cathal Boogerd and Leon Moonen have done a research on Evaluating the Relation between Coding Standard Violations and Faults Within and Across Software Versions.[9] According to that research, they have described three research questions regarding the point of view. The first question is “Do the releases with a higher violation density more fault-prone?” They have addressed this question using Cross-release analysis. However, they couldn’t find any relation between version releases with higher violation density apart from some individual rules. Next question is “Do the files or modules with a higher violation density more fault-prone?” To answer this question, they have used In-release analysis and by this analysis, they found a relation between violation density and number of defects of a software project. The last question is “Do the lines with violations more likely to point to faults than lines without?” The line-based analysis is used to investigate this problem. The result found was that adherence to a complete coding standard without customization may increase the probability of faults.

Moreover, in order to prevent coding standard violations and to keep the quality of code, pair programming method is used.[10] In this method two developers cross-checked their codes in order to find defects. However,this method also has some drawbacks. Since this is a manual test, there is a much probability to make mistakes.

2.3 Usage of Pattern analysis

Usually, the pattern analysis is used to detect patterns automatically from the same data source and make predictions of upcoming patterns from the same data source.[11] These data can be taken by many forms such as text, image, transaction history records, genome sequence, and family tree. By the way, there are some applications like anti-virus software and instruction detection systems which used to improve the data security over the internet using string matching techniques.[12] Apart from those applications analysis of protein expression, analysis of chemical formulas,[13] gene identifications, sequence analysis and evolutionary biological studies are commonly used for string matching techniques. Other than that, many other scientific subjects like Artificial Intelligence, Image Processing,[14] Computational Linguistics, Sound systems used string matching algorithm to implement their logic and tools.

In the year 2010, S. Harris, A. Averbuch, and N. Rabin found a fast compact prefix encoding for pattern matching in limited resources devices.[15] There they had an address to search and decompress textual data in a machine or device that has limited memory. They have used a binary representation of an integer as the prefix to encode the text.

Ofir Pele and Michael Werman presented a method for robust real-time pattern matching.[16] They introduced a group (collection) of image distance measures, the Image Hamming Distance set. There are four main components robust to occlusion, small geometrical transforms, light changes, and non-rigid deformations. Then they have presented a novel Bayesian framework for sequential hypothesis testing on finite populations. Based on that framework, they have designed a sampling algorithm to optimal rejection or acceptance. Using this algorithm, they can quickly determine whether two images have similarities with respect to a number of the image Hamming distance set. They have also presented a fast framework that can design a near optimal sampling algorithm. The test results of their experiment showed excellent performance.

2.4 Encoding and Decoding in String matching

The purpose of encoding is to convert some information from one format to another format or code.[17] The encoder can be a device, circuit, transducer, software program, algorithm or a person. Encoding is used to standardization, secrecy, speed up, security, or saving space by shrinking the size. The encoder encrypts information using a combination of logic and the decoder is used to retrieve back the original information from encoded data by using the same logic.

2.5 Usage of encoding and decoding

Encoding has a different meaning from coding. Cording is the set of instruction that tells the computer what to do. This means that entire computer programming systems are based on cording since computers have no freewill without explicit instruction.[18] All of these applications we do with computers like playing games, sending emails, search for something on Google, write a word document, take a selfie by a smart phone which is also a mini computer, talk with our family member on Skype, watch a movie on VLC player or buy something from eBay, are software written in cords.

The terms "encoding" and "decoding" are rapidly used in reference to the processes of analogue to digital conversions like in radio conversations and digital to analogue conversion like in television conversations.[19] In addition, these terms can also apply to any form of data, including text, images, audio, video, multimedia, computer programs, signals in sensors, telemetry, and control systems.

Encoding is rapidly used in computers since it is a process of putting a sequence of characters like letters, numbers, punctuation and certain symbols into a specialized format for efficient transmission or storage.[20] Decoding is used in computers to convert an encoded format back into the original sequence of characters. Both Encoding and Decoding are used in industries like data communications, networking, and storage. These terms are exceptionally using full in wireless communication systems.

Since numerous encoding and decoding are exist, there are few specialized coding systems which are used only by specialized groups of people such as Amateur radio operators, for example. The oldest code of all, originally used in the landline telegraph systems In 21st century.[21] Also in digital electronic projects, these encoding and decoding systems play an important role. Generally, these encoding systems are frequently used in the telecommunication, networking and transfer data from one end to the other end. In the same way encoding and decoding also used in the digital domain for easy transmission of data, placed with the codes and then transmitted.

2.6 Exiting Tools

Reshaper is one of the best tools developed by JetBrains team which helps developers to manage their coding standard.[22] It allows the developers to configure them manually or by default Reshaper which has been widely accepted in conventions and best practices. With this tool, violations of the code, style is detected with code inspections and it can be fixed with quick-fixes or code cleanup. However, the Reshaper needs to be configured manually for each developer. It doesn’t contain a global configuration mechanism.

CodeRush is another tool present by DevExpress team.[23] Apart from Reshaper, CodeRush can be considered as the main alternative to the Reshaper. There is no significant difference to be identified between these tools. The developer can experience the enhanced refactoring and productivity plugin which will extend the inbuilt functionality of Microsoft Visual Studio by using CodeRush. However, we couldn’t find any feature to maintain a common set of coding standard guideline by using this tool too.

The Telerik team developed a tool that can be integrated with Visual Studio 2005, 2008, 2010 and 2012 as an add-on.[24] It provides on-the-fly code analysis and error checking, refactoring codes and smart code navigation which can boost the Microsoft .NET framework based development productivity. Mainly, JustCode has a cross-language engine. Therefore, it can be used to develop C#.NET, VB.NET, ASP.NET, XAML, Razor, HTML, Java Script and CSS. However, it doesn't provide any coding standard violation detection mechanism.

There are some other features provide by Microsoft Visual Studio[25] as inbuilt features including enhanced support for multi-targeting, parallel programming and debugging, call hierarchy of methods, XSLT profiling and debugging, quick search, XSD designer and UML Designer. However, it only provides features to fast development and it doesn't provide any coding standard violation detection mechanism.

The Whole Tomato Software team present Visual Assist[26] as a plug-in for Microsoft Visual Studio. Main features of this plug-in are IntelliSense and syntax highlighting and it also enhances the code suggestion. Apart from the above features, it will provide refactoring commands and support for comments including spell checking suggestions. The Visual Assist will be able to detect basic syntax mistakes like the use of undeclared variables, code after return and data type mismatch. From the year 2017, Visual Assist also implemented for supporting Visual C++ 6.0 through the most of Visual Studio versions, including Visual

Studio Community edition version 2017 and Visual Studio version 2017. However, Visual Studio Express edition has some problems in third-party extensibility and it uses a separate extensibility model and therefore Visual Assist cannot be installed for the Studio Express editions.

VSCommands[27] is a tiny tool developed by a group of software developers. They believe coding should be easy with correct tools. The first release of VSCommands was in 2010 and VSCommands has been downloaded more than 2,000,000 times. Also the VSCommands tool has been used by thousands of developers in the world and it can increase developer productivity. VSCommands has two versions namely VSCommandslite (free version) and VSCommands pro (paid version).

In addition, there are several tools proposed in the literature in the direction of software security violation detection[28] and code clone detection[29]. However, these tools focus on software quality in general.

2.7 Challenges and Gaps Identified by the Literature Review

For the method used to maintain coding standard of the above exciting tools we have identified in our literature reviewis limited to specific programming language. Mainly all the above tools are focusing on identifying the syntax errors and logical errors and speed up the codings by providing interactions and code refactoring facility. Therefore, the coding standard violation detection is not properly addressed by those tools. Table-1 shows a summary of the gaps we have identified.

Exiting tool	**Gaps in detect coding standard violation**
Reshaper	Can configure to detect coding standard violation rules, however, there is no feature to manage standard using a common reference
CodeRush	Same as the Reshaper, only individual configurations are allowed
Telerik Add-on	Does not support to manage coding standard violations
VS Inbuilt functions	Does not support to manage coding standard violations
Visual Assist	Does not support to manage coding standard violations
VSCommands	Does not support to manage coding standard violations

Table 1 – Exiting tools and gaps

2.8 Problem Definition

As we mentioned in the previous highlights of journals related to coding standard violations, we can clearly identify the un-standard coding styles without best practices that can lead to the project failures. We have identified a few exiting tools which can help developers to do the implementation through a pre-defined standard. However, we could not find a common coding standard guideline from any of these tools except individual configuration.

2.9 Summary

This chapter discussed the evaluation of the coding standard violation and their effects with past, current and future challenges of software development domain. We also identified our research problem as the difficulty of software maintaining and enhancement due to unstructured coding patterns of individual developers. In addition, we identified developing a coding standard violation detection tool which will address the issue. Next chapter will describe our approach to solve the problem.

Chapter 3

3 Coding Standard Violation Detection by Pattern Analysis

3.1 Introduction

In the previous chapter we discussed a literature review of coding standard violation detection. This chapter we presents our approach to addressing the problem of cording standard violation in software development. For this purpose, we describe our hypothesis input-output process users and features in our approach.

3.2 Hypothesis

Our hypothesis is that the software failures in enhancement and future development, coding standard violations should be reduced by using pattern analysis mechanism. This hypothesis was inspired by looking at the importance of structuring and ordering in living and non-living things in the world.

When we consider about two cities in two different countries where one is from 3rd world country and other one from developed country, we can inspire very important facts about ordering and structuring things. That is similar to software development too. Without a proper standard, it is very difficult to handle the software development life cycle even though the software is working properly at the moment. The request for a change will be the beginning of the end of that software’s life cycle, if it doesn’t have a proper standard.

3.3 Input

As the input for the proposed tool required a coding standard guideline through the online reference and the source code to analyze. The online reference will be a web URL which contains a code segment with the standard guideline. Appendix-A shows the sample code segment.

3.4 Output

The Result of the pattern analyzing between standard guideline and the developer's codes will be the output. The output will be display on the screen. If any coding standard violation was detected, then the output contains a description of the violation and suggestions.

3.5 Process

When the inputs are submitted, this tool will capture the standard coding samples from the online reference. That will be stored in a database. Then it will generate encoded character stream which includes the behavior of the coding, captured from the guideline. Figure-3 will show the diagram of the entire process. Appendix-A shows a sample of the online source and its encoded character stream. Then the developer’s source code also converted to a character stream using the same encoding method.

In order to generate the encoded character stream for program code, we need to abstract all the properties and behaviors from the guideline program. Therefore, we have decided a list of common attributes and behaviours for a specific program. Appendix-B will show the detailed list. Generated character stream contains two digits to explain its attribute or behavior. Now, we have two character streams for the guideline program and the developer's program. Using this output, we will analyze the pattern and identify the differences of the two programs. The deviation of the developer program from the guideline program will be identified and then it is possible to capture the coding standard violations and acceptable differences.

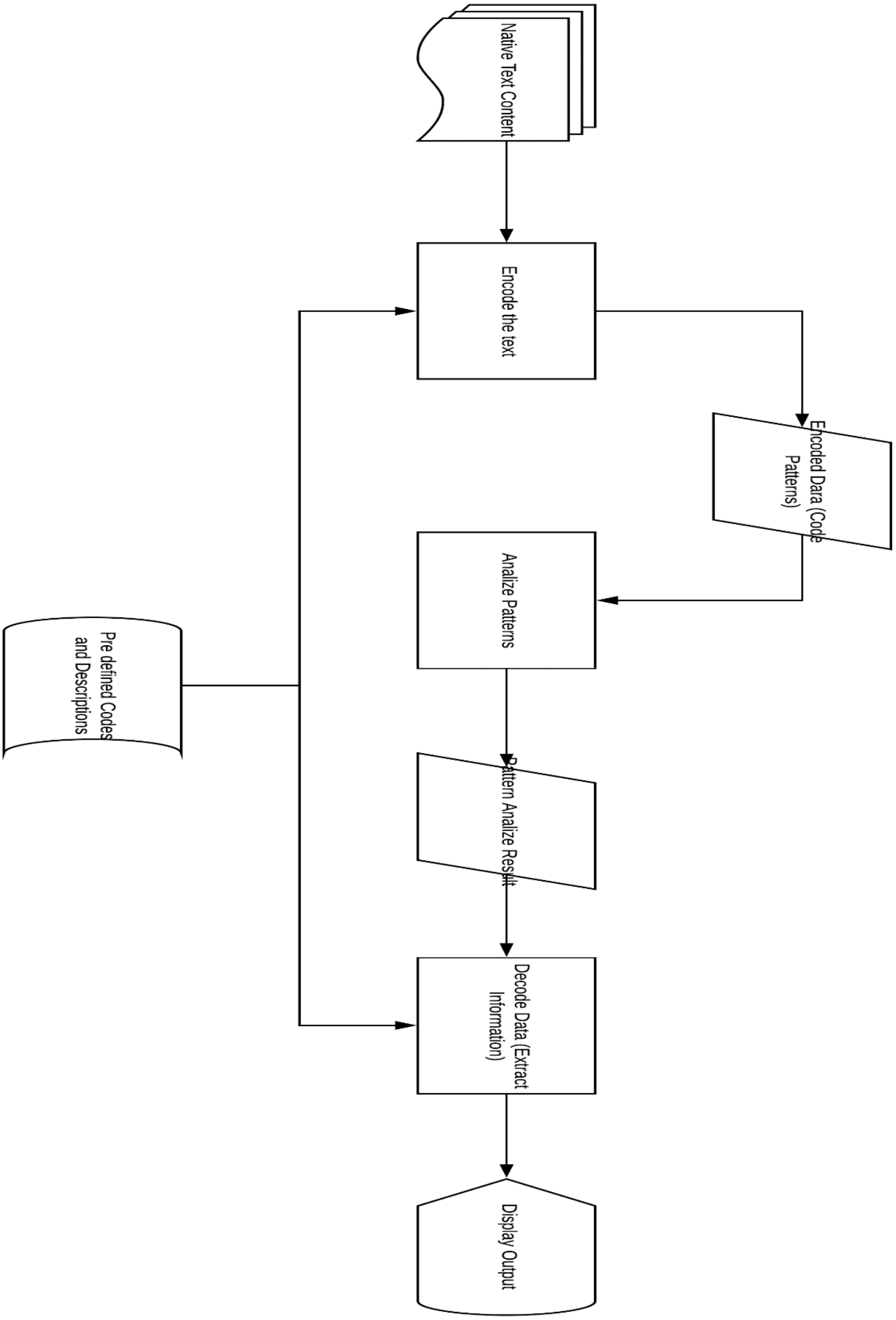

Native Text Content
Encode the text
Encoded Dara (Code Patterns)
Analize Patterns
Pre defined Codes and Descriptions
Pattern Analize Result
Decode Data (Extract Information)
Display Output

Figure 3 – Proposed Pattern Analysis method

3.6 Features

The proposed tool can be integrated with the development IDE and detect coding standard violations using a common standardguideline.

3.7 Summary

This chapter we discussed our hypothesis to solve the research problem, and also discussed the inputs and output of this solution. With this hypothesis, we propose a solution to our main research problem of coding standard violation. Next chapter we will discuss the design which will explain the high-level architecture of the solution.

4 Design

4.1 Introduction

Chapter3 we described our Approach to solve the research problem. This chapter presents a detail view of the design phase, in order to provide a brief description of our solution architecture. Here we will discuss the components and their individual roles, front end and backend designs to solving the problem of cording standard violation in software development.

4.2 High-Level Architecture of the proposed solution

The proposed tool has three major components as the data capture module, the database module, and the data analysis module. Data capture module contains capture data from the online reference and capture data from the user inputs. Database module is used to store the standard guideline. Analysis module used to analyze both user and guideline and identify the deference.

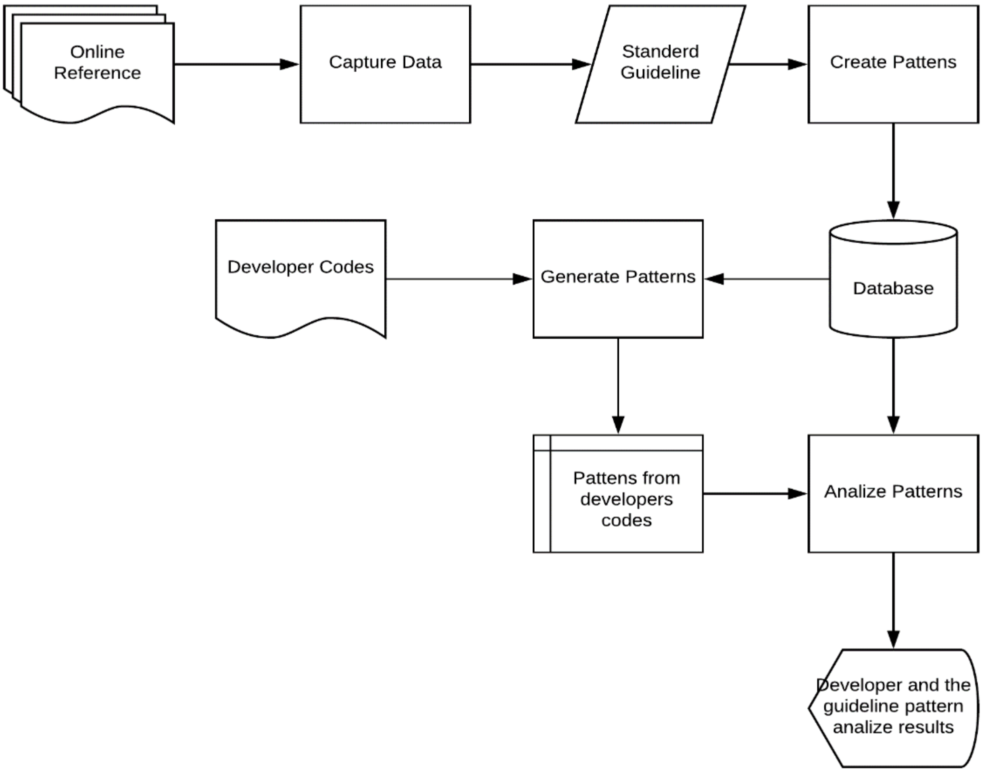


Figure 4-High-Level Architecture

- Capture Data
  This module is used to extract the guideline from an online reference. It accepts an URL which contains the guideline code as the input. Also, this module will generate the patterns according to the pre-defined attribute description.

- Database Module
  This module will keep the master data tables for the system and stored procedures used to implement pattern marching algorithm.

- Analysis module
  The analysis module is used to identify the coding standers violations using native pattern searching algorithm

4.3 Database Design

In the database design step, we have identified the main database requirement as in Figure 7. We need a suitable database to store our code description and online reference patterns to develop this system. In order to develop backend logic, we create a set of data tables as in Table 2

Table Name	Description
Code Line	Used to store the guideline code line by line with its relevant pattern generated by the proposed tool
Code Master	Used to store two digit codes and their descriptions
Keywords	Used to store specific keywords

Table 2- Database Table and Purpose

4.4 Backend programming

When developing the backend programming, all the variables and functions will be named according to the framework. There is a logical relationship between each and every function and variables.

4.5 Summary

This chapter we have discussed the Design of the proposed solution. Here we have identified the objects need to be implemented to achieve our goal. Next chapter will describe the implementation of the proposed solution.

Chapter 5

5 Implementation

5.1 Introduction

The previous chapter explains the overall design of the proposed solution. This chapter will give a brief explanation about the implementation (database, front end, and back end) of each module with actual interfaces and codes which we identified in chapter 4. Also, this chapter will give a brief description about the tools and technologies used to implement the proposed solution.

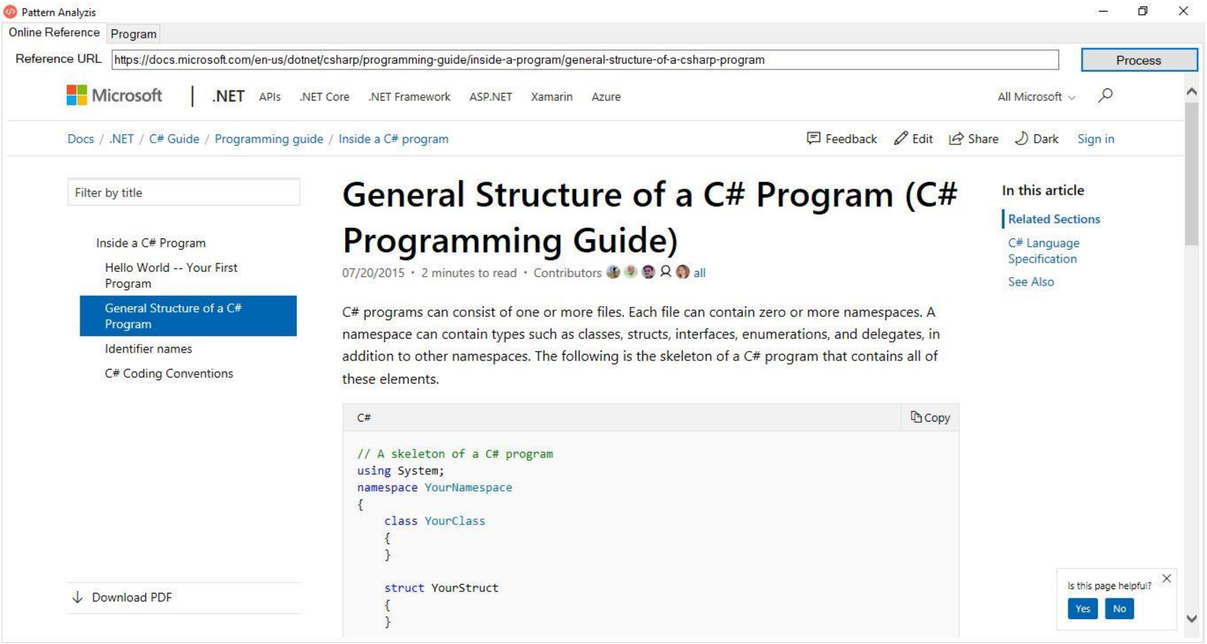
5.2 Tools and Technologies used to implement the proposed solution

MS SQL Server[28] also known as Microsoft SQL server is used to present the database role of the proposed framework. Microsoft SQL Server is a relational database management system developed by Microsoft to be used to manage and store information. Using MS SQL we create our data table's stored procedures and functions related to the proposed framework.

As a programming language, we used C#.NET[29] to design and implement the sample project. C#.NET is a framework which contains Microsoft standard class Libraries. Since we are developing a windows form application, C# is the best language for writing Microsoft .NET applications. C# is one of the best languages which support the object-oriented concept (OOP) Abstraction, Encapsulation, Polymorphism and Inheritance. It also provides support to rapid application development.

5.3 Fetching the standard guideline from online reference

One of the major objectives of the proposed solution captures the standard guideline from an online reference. For the purpose of capturing the standard guidelines from an online reference, we have implemented our tool as below [Figure 5]. When the reference URL was submitted, the content will be loaded into the browser and then we identify the codes in the HTML content pages. Then the code extraction will be done. The code snippet for this function will be shown in Appendix-D.



Reference URL https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/general-structure-of-a-csharp-program

General Structure of a C# Program (C# Programming Guide)

07/20/2015 • 2 minutes to read • Contributors all

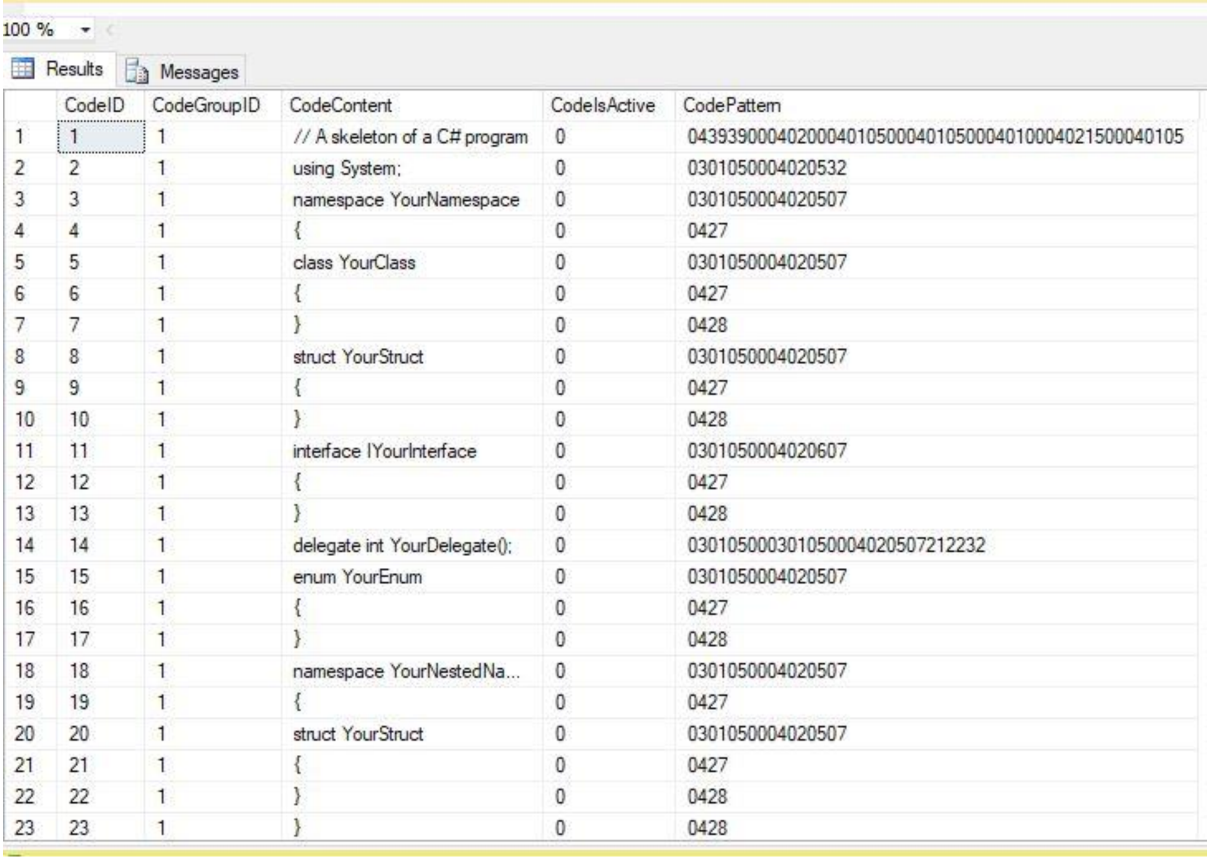
C# programs can consist of one or more files. Each file can contain zero or more namespaces. A namespace can contain types such as classes, structs, interfaces, enumerations, and delegates, in addition to other namespaces. The following is the skeleton of a C# program that contains all of these elements.

```
// A skeleton of a C# program
using System;
namespace YourNamespace
{
    class YourClass
    {
    }

    struct YourStruct
    {
    }
```

Figure 5 - Interface to fetching the guideline from an online reference

For each codeline we have captured from the online reference, will be stored in the database. In this step, we identify also the code patterns for those codelines. Figure-6 will show how the auto-generated patterns and code lines are stored in the database.

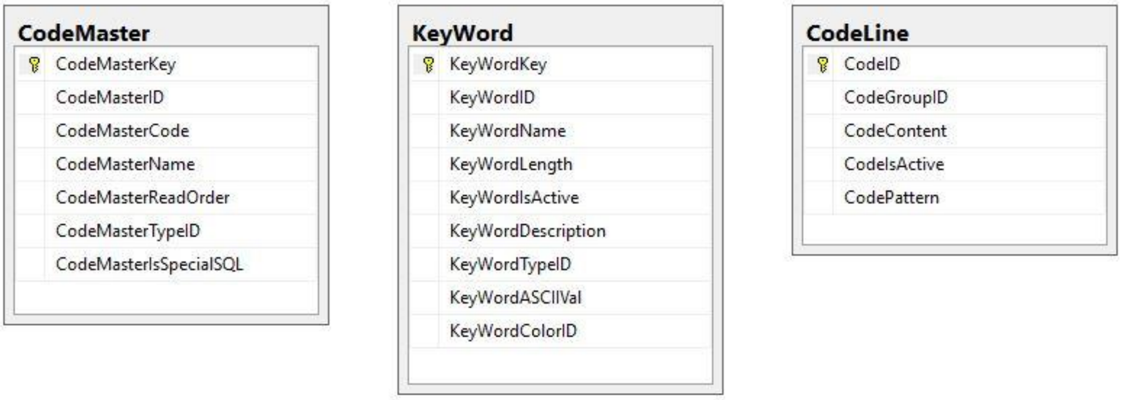


	CodeID	CodeGroupID	CodeContent	CodeIsActive	CodePattern
1	1	1	// A skeleton of a C# program	0	04393900040200040105000401050004010004021500040105
2	2	1	using System;	0	0301050004020532
3	3	1	namespace YourNamespace	0	0301050004020507
4	4	1	{	0	0427
5	5	1	class YourClass	0	0301050004020507
6	6	1	{	0	0427
7	7	1	}	0	0428
8	8	1	struct YourStruct	0	0301050004020507
9	9	1	{	0	0427
10	10	1	}	0	0428
11	11	1	interface IYourInterface	0	0301050004020607
12	12	1	{	0	0427
13	13	1	}	0	0428
14	14	1	delegate int YourDelegate();	0	030105000301050004020507212232
15	15	1	enum YourEnum	0	0301050004020507
16	16	1	{	0	0427
17	17	1	}	0	0428
18	18	1	namespace YourNestedNa...	0	0301050004020507
19	19	1	{	0	0427
20	20	1	struct YourStruct	0	0301050004020507
21	21	1	{	0	0427
22	22	1	}	0	0428
23	23	1	}	0	0428

Figure 6- Generated patterns for the online guideline

5.4 Database implementation

In order to generate patterns, we have used 3 physical data tables in Figure-7. Based on the requirement to keep exiting keywords in C#.NET language, we have to use the KeyWord table. CodeMaster table is used to keep the description of 2 digit codes which we used to encode the codelines. CodeLine table is used to store the guideline and its auto-generated patterns.



CodeMaster
CodeMasterKey
CodeMasterID
CodeMasterCode
CodeMasterName
CodeMasterReadOrder
CodeMasterTypeID
CodeMasterIsSpecialSQL

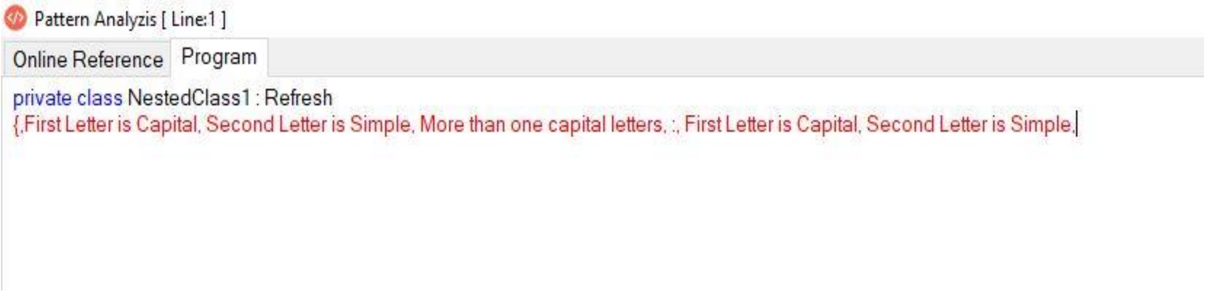
KeyWord
KeyWordKey
KeyWordID
KeyWordName
KeyWordLength
KeyWordIsActive
KeyWordDescription
KeyWordTypeID
KeyWordASCIIVal
KeyWordColorID

CodeLine
CodeID
CodeGroupID
CodeContent
CodeIsActive
CodePattern

Figure 7 - Set of database tables

5.5 Capture the developer code

We have developed an interface to capture the developer code. This interface is also used to display the result of the pattern analysis which will inform the violations of coding standard. Figure-8 shows the interface to capture developer code.



```
private class NestedClass1 : Refresh
{,First Letter is Capital, Second Letter is Simple, More than one capital letters, :, First Letter is Capital, Second Letter is Simple,
```

Figure 8 – Interface to capture developer codes

Once we compare the patterns in order to test our hypothesis, we need to collect sample data sets from an online reference and developer code. Sample code segments will be shown in Table-3

Online Reference	Generated Pattern	Developer Code	Generated Pattern
class YourClass	030004020507	public class MyTestClass	0300030004020507
		public class mytestclass	03000300040105
		public class Mytestclass	0300030004240624

Table 3 - Sample code segment generated by the proposed tool

5.6 Identify the violations using Pattern Analysis Mechanism

In this step, we have used Native Pattern Searching algorithm to compare patterns on both guideline codes and the developer code. The comparison should be done to find patterns on the guideline from the developer code as well as to find patterns on the developer code from the guideline. Because patterns can occur as Table-4

Guideline (Reference Code)	**Developer Code**
class YourClass	public class MyTestClass
030004020507	0300**030004020507**
private Int. variable	Int. number
0300**0300040105**	**0300040105**

Table 4 -Sample Pattern occurrence in both guideline and the developer code

Once we implement this algorithm, it is essential to cross-check patterns to detect pattern occurrences in both sides. If the pattern does not match in both sides, then it is identified as a coding standard violation. The Native Pattern Searching algorithm implementation code snippet will be shown in Appendix-E.

The following pseudocode will be demonstrating the Native pattern Search algorithm. Slide the pattern over the search, text one by one and check for matching parts. If a matching index is Identified, then it shifts by 1 again next to check for other subsequent matches.

```
txt = “AABAACAADAABAAABAA”;
pat =“AABA”;
    M = pat.Length; N = txt.Length;
    FOR i = 0 TO i <= N – M
      J =0
      FOR j = 0 TO j < M
if (txt[i + j] != pat[j])
          BREAK
if j == M
THEN  PRINT PATTER FOUND AT i
      j++
i++
```

5.7 Summary

In this chapter we have discussed the Implementation of the proposed solution. The next chapter will describe how the database and backend coding was implemented. Next chapter will show the evaluation of our solution to detect coding standard violation by pattern analysis mechanism.

Chapter 6

6 Evaluation

6.1 Introduction

The previous chapter shows the implementation of the proposed tool with a detailed review of database and coding implementation. In this chapter, we will present an evaluation of the test result against the primary objectives of the project. This chapter also provides our data collection and the results of the evaluation method.

6.2 Purpose of evaluation

The evaluation of the system has been carried out to check the achievements of the objectives of the project. For this purpose, we have considered materials to be tested which are included in the approach. As such, the evaluation is concern with input-output process users and features in connection with the hypothesis.

6.3 Evaluation method

First, we have created an evaluation form with a sample program that has some coding standard violations to cover a few coding standard rules according to the selected guideline. The sample program will be shown in Appendix-C

Then we have to contact 7 software engineers and asked them to go through a sample program code and identify the coding standard violations. The evaluation form will be shown in the Appendix-D section. After collecting the evaluation forms which were manually done by the selected software engineers we have summarized the results from each individual. A summary table will be shown in Table-5. Then we have identified that there can be three scenarios as in Figure-9.

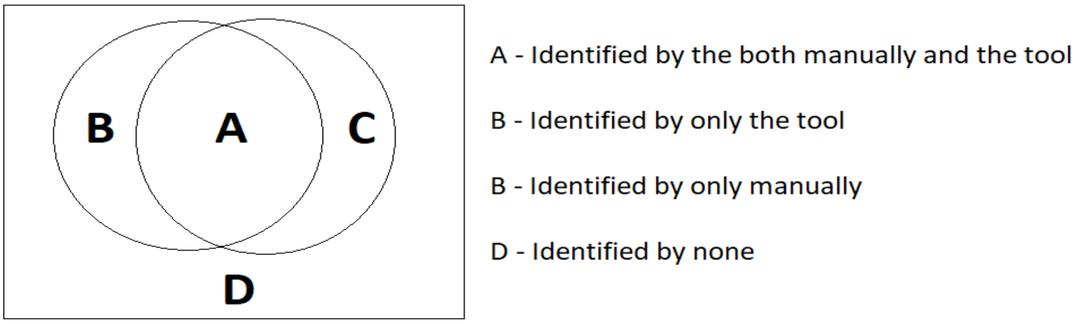

B
A
C
D
A - Identified by the both manually and the tool
B - Identified by only the tool
B - Identified by only manually
D - Identified by none

Figure 9 - Possiblescenarios to identify coding standard violations.

6.4 Data collection

The following Table 5presents the summary of Evaluation forms.

Software Engineer	Identified coding standard violations
Dananjaya Mathes Software Engineer Scienter Technologies PTE Service Experience 5 Years 0773525924	1. Declare classes and variables using specialcharacters 2. Create multiple instances to thesame class object 3. Interface name should begin with the letter "I"
S.G.A Chandrakumara Software Engineer Scienter Technologies PTE Service Experience 11 Years 0778151151	1. Classes and method names should be declared using Pascal case 2. Method argument and local variables should be declared using camel case 3. Cannot use underscore to declare to identifiers 4. Interface name should start with "I"
W.A.T Kaushalya Tech Lead Scienter Technologies PTE Service Experience 8 Years 0772531679	1. Unnecessaryif else statements 2. Underscore used to declare classes 3. The interface should start with "I" letter
Kalindu Kasun Software Engineer Scienter Technologies PTE Service Experience 6 Years 071624299	1. Variable cannot be declared with an underscore 2. The methodcannot be declared with an underscore 3. Class names can not contain numbers 4. The method should be declared with Pascal Case 5. The interface should be declared using "I" as a prefix
Dilan Semasinghe Senior Software Engineer Scienter Technologies PTE Service Experience 5 Years	1. Does not usemeaningful class variable and names for some classes 2. Does not used appropriate prefixes 3. Does not used appropriate pascal case and camelcase to declare variables methods and classes

Table 5 - Summary of Evaluation Forms

Here we can highlight the following coding standard violations.

- Special character underscore "_" used as the first letter of class declaration (identified by 5 software engineers)
- Special character underscore "_" used as the first letter of variable declaration (identified by 5 software engineers)
- Special character underscore "_" used as the first letter of method declaration (identified by 5 software engineers)
- Multiple object creation for the same instance (identified by 1 software engineer)
- Interface class does not declare with the letter "I" as a prefix (identified by 5 software engineers)
- Unnecessary else statement (identified by 1 software engineer)
- Method and class should be declared using Pascal case (identified by 3 software engineers)

- The local variable should be declared with camel case (identified by 3 software engineers)
- Class name contains numbers (identified by 1 software engineer)
- Unnecessary if else statement (identified by 1 software engineer)
- Does not usemeaningful names to class, method, and variables (identified by 1 software engineer)

Following Table 6 present the result of evaluating the same code using the proposed tool

Coding standard violation	**Is detected by the software engineers**	**Is detected by the proposed tool**	**No of Detection**
Special Characters used in class, method and variable declaration	YES	YES	5/5
Multiple object creation for the same instance	YES	YES	1/5
Interface class does not declare with the letter “I” as a prefix	YES	YES	5/5
Unnecessary else statement	YES	YES	1/5
Method and class should be declared using Pascal case	YES	YES	3/5
The local variable should be declared with camel case	YES	YES	3/5
Class name contains numbers	YES	NO	1/5
Does not usemeaningful names to class, method, and variables	YES	NO	1/5

Table 6 - Result of Evaluation Form

6.5 Evaluation of results

We have highlighted a new factor, when we check the code using Pair Programming there is more probability of missing the identification of some coding standard violations. Because the evaluation results show some experienced software engineers also could not notice some violations even the simple program.

6.6 Summary

Here we have discussed the evaluation of our solution to coming standard violation detection by pattern analysis. Our evaluation method shows the preface of the proposed solution. In the next chapter, we will discuss the conclusion of this research.

Chapter 7

7 Conclusion

7.1 Introduction

The previous chapter we discuss the evaluation of this project. There we've analyzed the results from various samples. This chapter will discuss the conclusion of this project and also further developments.

7.2 Overall Conclusion

Based on the evaluation in Chapter 7, the proposed solution can detect coming standard violations using their pattern analyzing mechanism. Therefore, the proposed tool can prevent coding standard violations. According to the result of evaluation forms, this tool clearly shows its performance identifying the coding standard violations detected by the selected group of software engineers.

We have highlighted a major point in our evaluation, which is about pair programming. The pair programming is used to keep the code quality and standard by cross-checking the codes by using two developers. However in this evaluation, we have provided a sample class to five experience software engineers. But only two violations out of six, identified by all of them and four other violations identified by a few of them. This shows pair programming can have some probabilities to make mistakes.

When manually evaluated a code, there can be some individually defined rules which are not in according to the standard. Therefore, we can conclude that, an automated coding standard detection system is required to prevent that issue and it should be able to configure using a common guideline.

In our evaluation, one software engineer was mention that there should not include numbers to the class names. However, that this point is arguable because some meaningful names also contain numbers. Area51, Zone24 and T56 can be taken as some example for the names with numbers. This point also related to the fact that we have previously mentioned.

7.3 Objective Wise Conclusion

Our main objective is to design a tool that can identify coding standard violations. We have successfully designed a tool to achieve that goal. Then we have implemented the tool we designed which can address the problem that has been identified in our literature review. We have evaluated the tool and we have implemented using five professional and experienced software engineers. Finally, we have published a thesis to illustrate our journey to implement the proposed tool to the end from the beginning successfully.

7.4 Limitation

In our evaluation form, one software engineer has mentioned that there should be meaningful names to class, methods, and variables. However, that particular feature has not been implemented in this tool since there is a technical difficulty to catch the meaning. By the way, none of the exiting tools provided that feature and assigning meaningful names is totally depend on the developer.

7.5 Further Works

As further works, we will develop this tool to identify meaningful words. In order to achieve that target, we should identify some technical differences between meaningful words and meaningless worlds. However, the concept of coding standards violation can be effectively explored with the help of software engineering analytics tools as proposed in [32] and [33].

Moreover, as the future development, the proposed pattern analysis mechanism can be used for grammar checking purpose for any language because the encoded pattern is independent of its original source.

8 References

[1] T. Cowling, “Model-driven development and the future of software engineering education,” in *Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on*, 2013, pp. 329–331.

[2] V. Chiew and Y. Wang, “A large-scale empirical study on the cognitive complexity of software,” in *CCECE 2010*, Calgary, AB, Canada, 2010, pp. 1–4.

[3] “Leading reasons for software project failure according to developers worldwide, as of 2015.” https://www.statista.com.

[4] C.-Z. Li, K.-H. Hsu, and G.-Y. Chen, “Discovering Aspects through Analyzing Code Changes in Software Development Histories,” 2015, pp. 297–302.

[5] A. Dearle, “Software deployment, past, present and future,” in *2007 Future of Software Engineering*, 2007, pp. 269–284.

[6] N. A. Razak and M. Ghazali, “Usability in software development: Frameworks comparison between IKnowU and user behavior analysis framework (UBAF),” in *Software Engineering (MySEC), 2011 5th Malaysian Conference in*, 2011, pp. 330–335.

[7] A. Carzaniga, A. Fuggetta, R. S. Hall, D. Heimbigner, A. Van Der Hoek, and A. L. Wolf, “A characterization framework for software deployment technologies,” DTIC Document, 1998.

[8] J. L. B.-J. Nelson Martínez-Araujo and Alejandro González-García1, “Software Reuse and Continuous Software Development: A Systematic Mapping Study,” *IEEE*, 2018.

[9] C. Boogerd and L. Moonen, “Evaluating the relation between coding standard violations and faultswithin and across software versions,” in *Mining Software Repositories, 2009. MSR’09. 6th IEEE International Working Conference on*, 2009, pp. 41–50.

[10] M. Nawahdah and D. Taji, “Work in progress: Investigating the effects of pair-programming on students’ behavior in an advanced computer programming course,” in *2015 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, Zhuhai, China, 2015, pp. 157–160.

[11] Prof. Bennett, “Math Model of Learning and Discovery,” http://www.rpi.edu/~bennek/class/mmld/talks/lecture2-05.ppt, 14-Feb-2005.

[12] X. Li and Q. Wen, “A fast multi-pattern matching algorithm for anti-virus scanning,” in *2011 4th IEEE International Conference on Broadband Network and Multimedia Technology*, Shenzhen, China, 2011, pp. 42–45.

[13] A. Yamaguchi, Y. Yamamoto, J.-D. Kim, T. Takagi, and A. Yonezawa, “Discriminative Application of String Similarity Methods to Chemical and Non-chemical

Names for Biomedical Abbreviation Clustering," in *2011 IEEE International Conference on Bioinformatics and Biomedicine*, Atlanta, GA, USA, 2011, pp. 544–549.

[14] Y. Watanabe and K. Takahashi, "A fast structural matching and its application to pattern analysis of 2-D electrophoresis images," in *Proceedings 1998 International Conference on Image Processing. ICIP98 (Cat. No.98CB36269)*, Chicago, IL, USA, 1998, vol. 3, pp. 804–808.

[15] S. Harrusi, A. Averbuch, and N. Rabin, "A Fast Compact Prefix Encoding for Pattern Matching in Limited Resources Devices," in *2010 Data Compression Conference*, Snowbird, UT, USA, 2010, pp. 533–533.

[16] O. Pele and M. Werman, "Robust Real-Time Pattern Matching Using Bayesian Sequential Hypothesis Testing," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 8, pp. 1427–1443, Aug. 2008.

[17] G. M. Landaut and S. Skiena, "Matching for Run-Length Encoded Strings," p. 9.

[18] J. Heer and M. Agrawala, "Software design patterns for information visualization," *IEEE Trans. Vis. Comput. Graph.*, vol. 12, no. 5, pp. 853–860, 2006.

[19] G. R. Higgie and A. C. M. Fong, "Efficient encoding and decoding algorithms for variable-length entropy codes," *IEE Proc. - Commun.*, vol. 150, no. 5, p. 305, 2003.

[20] M. A. El Affendi and K. H. S. Al Rajhi, "Text encoding for deep learning neural networks: A reversible base 64 (Tetrasexagesimal) Integer Transformation (RIT64) alternative to one hot encoding with applications to Arabic morphology," in *2018 Sixth International Conference on Digital Information, Networking, and Wireless Communications (DINWC)*, Beirut, 2018, pp. 70–74.

[21] R. W. P. King, "Electric fields induced in cells in the bodies of amateur radio operators by their transmitting antennas," *IEEE Trans. Microw. Theory Tech.*, vol. 48, no. 11, pp. 2155–2158, Nov. 2000.

[22] JetBrains Team, "Resharper," *Resharper*. [Online]. Available: https://www.jetbrains.com/resharper/.

[23] DevExpress Team, "CodeRush," *CodeRush*, 01-Aug-2018. [Online]. Available: https://www.devexpress.com/products/coderush/.

[24] Telerik, "JustCode," *JustCode*. [Online]. Available: https://www.telerik.com/. [Accessed: 06-Aug-2018].

[25] Microsoft, "Visual Studio," *Visual Studio*. [Online]. Available: https://visualstudio.microsoft.com/.

[26] Whole Tomato Software, "Visual Assist," *Visual Assist*, 06-Oct-2018. [Online]. Available: https://www.wholetomato.com/.

[27] Squared Infinity, “VSCommands,” *VSCommands*. [Online]. Available: https://marketplace.visualstudio.com/items?itemName=SquaredInfinityJarekKardas.VSCommands14forVisualStudio2015.

[28] Wijesiriwardana, C., & Wimalaratne, P. (2017, May). On the detection and analysis of software security vulnerabilities. In 2017 International Conference on IoT and Application (ICIOT)(pp. 1-4). IEEE.

[29] Wijesiriwardana, C., & Wimalaratne, P. (2017, November). Component-based experimental testbed to faciltiate code clone detection research. In 2017 8th IEEE International Conference on Software Engineering and Service Science (ICSESS) (pp. 165-168). IEEE.

[30] Microsoft, “Microsoft SQL Server,” *Microsoft SQL Server*, 10-Oct-2018. [Online]. Available: https://www.microsoft.com/en-us/sql-server/sql-server-2016. [Accessed: 10-Oct-2018].

[31] Microsoft, “Microsoft C#.NET,” *Microsoft C#.NET*, 10-Oct-2018. [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/getting-started/introduction-to-the-csharp-language-and-the-net-framework. [Accessed: 10-Oct-2018].

[32] Wijesiriwardana, C., & Wimalaratne, P. (2019). Software Engineering Data Analytics: A Framework Based on a Multi-Layered Abstraction Mechanism. IEICE Transactions on Information and Systems, 102(3), 637-639.

[33] Wijesiriwardana, C., & Wimalaratne, P. (2018). Fostering Real-Time Software Analysis by Leveraging Heterogeneous and Autonomous Software Repositories. IEICE TRANSACTIONS on Information and Systems, 101(11), 2730-2743.

9 Appendix-A

the sample code segment captured from the URL https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/general-structure-of-a-csharp-program

Sample code from the online reference	Encoder pattern

// A skeleton of a C# program
using System;
namespaceYourNamespace
{
public classYourClass : ISomeClass
    {
    }
 	
structYourStruct
    {
    }
 	
interfaceIYourInterface
    {
    }
 	
delegateintYourDelegate();
 	
enumYourEnum
    {
    }
 	
namespaceYourNestedNamespace
    {
structYourStruct
        {
        }
    }
 	
classYourMainClass
    {
staticvoidMain(string[] args)
{
//Your program starts here...
        }
    }
}
	Keyword, Space, Keyword, Space, Not a keyword,First letter is capital,the Second letter is not capital, Space, Colen, Space,Not a keyword,theFirst letter is capital,the Second letter is capital,Brackets (0011001102030411221102030599)

10 Appendix-B

Sample two digits code and its description

Code	Description
00	White Space
01	First Letter is Simple
02	First Letter is Capital
03	Key Word
04	Not a Key Word
05	The second Letter is Simple
06	The second Letter is Capital
07	More than one capital letters
08	Only one capital letter
09	Only the first two capital letters
10	First two capital letters and more than one other capital letters
11	`
12	~
13	!
14	@
15	#
16	$
17	%
18	^
19	&
20	*
21	(
22	)
23	-
24	_
25	=
26	+
27	{
28	}
29	[
30	]
31	:
32	;
33	'
34	"
35	,
36	<
37	>

38	.
39	/
40	\
41	?
42	|

11 Appendix-C

```
using System;
usingSystem.Collections.Generic;
usingSystem.Linq;
usingSystem.Text;
usingSystem.Threading.Tasks;

namespacePaternAnalyzis
{
publicclass_EvaluationProgram
    {
privatestaticintcurrentIndex;
privatestaticint_nextIndex;
public _EvaluationProgram()
        {
currentIndex = 0;
            _nextIndex = 1;
        }
privatevoidTestMethod()
        {
intidNext =
newNestedClass1().GetNextIndex();

NestedClass1obj = newNestedClass1();
intidCurrent = obj.GetCurrentIndex();
        }
privateclassNestedClass1 : IRefresh
        {
internalintGetNextIndex()
            {
return _nextIndex;
            }
internalintGetCurrentIndex()
            {
returncurrentIndex;
            }
publicint Refresh()
            {
if (currentIndex == 1){
return 1;
                }
else
                {
return 0;
                }
            }
```

```
publicvoid cancel()
            {
if (currentIndex == 1)
                {

                }
else
                {

                }
            }
        }
privateclassnestedClass2 : Multiply
        {
publicint Multiplication()
            {
return _nextIndex> 0 ? _nextIndex *
currentIndex : 1;
            }
        }

    }
publicinterfaceMultiply
    {
int Multiplication();
    }

publicinterfaceIRefresh
    {
int Refresh();
void cancel();
    }
}
```

12 Appendix-D

Code snippet to capture the online guideline from aURL

```
privatevoidReadCode(string code)
        {
code = Regex.Replace(code, @"<[^>]*>", String.Empty);
string[] lines = code.Split(new[] { "\r\n", "\r", "\n" }, StringSplitOptions.None);
CodeModel c = newCodeModel();
Service<CodeModel>obj = newService<CodeModel>();
intCodeGroupID = obj.GetNextGroupID<CodeModel>(c).CodeGroupID;
for (int i = 0; i <lines.Length; i++)
            {
c.CodeGroupID = CodeGroupID;
c.CodeContent = lines[i];
if (!obj.InsertToCodeLine(c))
                {
//insert error
                }
            }
        }



privatevoid webBrowser1_DocumentCompleted(object sender,
WebBrowserDocumentCompletedEventArgs e)
        {
HtmlDocumenthtmlDocument = webBrowser1.Document;
HtmlElementCollectionhtmlElementCollection = htmlDocument.All;

List<HtmlElement>eList = htmlElementCollection.Cast<HtmlElement>().ToList();

List<HtmlElement>eListOut = (from a ineList
wherea.TagName.ToUpper().Contains("CODE")
select a).ToList();

foreach (HtmlElement elm ineListOut)
            {
ReadCode(elm.InnerHtml);
            }
        }
```

Generating patterns

```
set@cols=REPLACE(@cols,'{',' { ')
       set@cols=REPLACE(@cols,'}',' } ')
       set@cols=REPLACE(@cols,')',' ) ')
       set@cols=REPLACE(@cols,'(',' ( ')
       set@cols=REPLACE(REPLACE(@cols,CHAR(13),' '),CHAR(10),' ')
       set@cols=REPLACE(REPLACE(REPLACE(ltrim(@cols),' ',' %'),'% ',''),'%','')

       Select*,0asgIDinto#tfrommaster.dbo.split(@cols,' ')

       createtable#ob(idintidentity(1,1),obIDint)
       createtable#obtocb(idintidentity(1,1),obIDint,cbIDint)
       declare@kint=1,
                     @iint=1,
                     @lastOBindexint,
                     @itemvarchar(max)
       while@k<=(selectcount(*)from#t)
```

```
    begin
        select@item=itemsfrom#twherenameindex=@k
        if@item='{'
        begin
            insertinto#obselect@k
            set@lastOBindex=@k
        end
        elseif@item='}'
        begin
            insertinto#obtocb
            select@lastOBindexob,@kcb

            deletefrom#obwhereobID=@lastOBindex
            set@lastOBindex=(selecttop 1 obIDfrom#oborderbyiddesc)
        end
        set@k+=1
    end

    declare@rangetable (IDintidentity(1,1),startIDint,endIDint)
    insertinto@range
    selectobID,cbIDfrom#obtocborderbyobID

    declare@i1int=1,@startIDint,@endIDint

    while@i1<=(selectcount(*)from@range)
    begin
        select@startID=startID,@endID=endID
        from@rangewhereID=@i1
        update#tsetgID=@startID
        wherenameindexbetween@startIDand@endID
        set@i1+=1
    end

    selectt.*,ROW_NUMBER()over(partitionbyt.gIDorderbyt.nameindex)pID,b.cbID
    into#a
    from#ttleftouterjoin
    #obtocbbont.nameindex=b.obID
    orderbyt.nameindex

    select*,ROW_NUMBER()over(orderbynameindex)newOdr,0
seqinto#bfrom#aorderbynameindex
    declare@btable(idintidentity(1,1),gidint,seqint)
    declare@prvGidint=0,@currGidint=0,@seqint=1

    set@i=1
    while@i<=(selectcount(*)from#b)
    begin
        select@currGid=gIDfrom#bwherenameIndex=@i
        if(@currGid<>@prvGid)
        begin
            set@seq+=1
            insertinto@bselect@currGid,@seq
            set@prvGid=@currGid
        end
        else
        begin
            insertinto@bselect@currGid,@seq
        end
        set@i+=1
    end
    updateb2setb2.seq=b1.seqfrom@bb1innerjoin#bb2onb2.nameIndex=b1.id
    declare@linetable (idintidentity(1,1),linevarchar(max))
    set@i=1
    set@seq=1
    declare@gIdint,@itmvarchar(max),@strvarchar(max)='',@maxPIDint,@pidint,@prvHead
ervarchar(max)=''
    while@i<=(selectcount(*)from#b)
    begin
        select@gId=gID,@itm=items,@pid=pID,@seq=seq
        from#bwherenewOdr=@i
        select@maxPID=max(pid)from#bwheregID=@gId
        if@gId=0
        begin
            set@str+=' '+@itm
            ifcharindex(';',@itm)>0
            begin
                insertinto@lineselect@str
                set@str=''
            end
            elseif@pid=@maxPID
            begin
                insertinto@lineselect@str
            end
            set@str=LTRIM(@str)
        end
        else
        begin
            declare@headervarchar(max)=''
            ;withtas(

    selectitemsfrom#bwheregID=@gIdanditemsnotin('{','}')andseq=@seq
            )select@header+=stuff((select' '+items+' '
            fromtforxmlpath(''),type).value('.','nvarchar(max)'), 1, 1,'')
            ifisnull(@header,'')<>''
            begin
                if@prvHeader<>@header
                begin
                    insertinto@line
                    Selectltrim(items)frommaster.dbo.split(@header,';')
                    set@prvHeader=@header
                end
            end
        end
        set@i+=1
    end

    selectlinefrom@linewherelinenotlike'%;'orderbyid

    droptable#t
    droptable#obtocb
    droptable#ob
    droptable#a
    droptable#b
```

13 Appendix-E

Code snipt of the Native Pattern Search Algorithm

```
declare@textvarchar(max),@linevarchar(100)='public void _class _class'
--exec GetPatternForCodeline @line,@text output

set@line=' '+@line+' '
set@line=REPLACE(@line,'{',' { ')
set@line=REPLACE(@line,'}',' } ')
set@line=REPLACE(@line,')',' ) ')
set@line=REPLACE(@line,'(',' ( ')
set@line=REPLACE(REPLACE(@line,CHAR(13),' '),CHAR(10),' ')
set@line=REPLACE(REPLACE(REPLACE(ltrim(@line),' ',' %'),'% ',''),'%','')

SelectnameIndex,LTRIM(RTRIM(items))code,''aspattern,''errinto#tblfrommaster.dbo.split(
@line,' ')
altertable#tblaltercolumnpatternvarchar(max)
altertable#tblaltercolumnerrvarchar(max)

declare@zint=1,@cdvarchar(500),@txvarchar(max)
while@z<=(selectcount(*)from#tbl)
begin
       set@cd=''set@tx=''
       select@cd=codefrom#tblwherenameIndex=@z
       execGetPatternForCodeline@cd,@txoutput
       update#tblsetpattern=@txwherenameIndex=@z
       set@z+=1
end

declare@inint=1,@userPatternvarchar(max)
while@in<(selectcount(*)from#tbl)
begin
       set@userPattern=''
       select@userPattern=patternfrom#tblwherenameIndex=@in

       -- Split the codeline, get the list of the keywords, filter it
              declare@tbltable(
              idintidentity(1,1),
              CodePatternvarchar(500),
              keyword1varchar(500),
              keyword2varchar(500),
              keyword3varchar(500),
              result1int,
              result2int
              )

              --declare @userPatternvarchar(max)=@text
              ;withas(

       selectc.CodeID,c.CodeContent,c.CodePattern,k.KeyWordNamekeyword1,len(k.KeyWordN
ame)+2 start,
              len(c.CodeContent)-len(k.KeyWordName)-1 length
              from[dbo].[CodeLine]cinnerjoin[dbo].[KeyWord]kon
              k.KeyWordName=substring(c.CodeContent,1,len(k.KeyWordName))
              ),mas(

       selectt.CodeID,t.CodeContent,t.CodePattern,t.keyword1,k.KeyWordNamekeyword2,
              len(k.KeyWordName)+len(t.keyword1)+2 start,len(t.CodeContent)-
(len(k.KeyWordName)+len(t.keyword1))-1 length
              fromtinnerjoin[dbo].[KeyWord]kon
       k.KeyWordName=substring(substring(t.CodeContent,t.start,t.length),1,len(k.KeyWo
rdName))
              wheret.length>1
              ),jas(

       selectm.CodeID,m.CodeContent,m.CodePattern,m.keyword1,m.keyword2,k.KeyWordNamek
eyword3
              fromminnerjoin[dbo].[KeyWord]kon

       k.KeyWordName=substring(substring(m.CodeContent,m.start,m.length),2,len(k.KeyWo
rdName))
              wherem.length>1
              ),las(
              selectCodePattern,keyword1,keyword2,keyword3,
              PATINDEX('%'+
              ((SELECTSTUFF((
              SELECTpattern+'00'
              from#tblwherenameIndexin(1,2)
              orderbynameIndex
              FORXMLPATH(''),TYPE).value('.','NVARCHAR(MAX)'), 1, 0,'')))
                     +'%',CodePattern)result1,-- index 1

              PATINDEX('%'+CodePattern+'%',((SELECTSTUFF((
              SELECTpattern+'00'
              from#tblwherenameIndexin(1,2)
              orderbynameIndex
              FORXMLPATH(''),TYPE).value('.','NVARCHAR(MAX)'), 1, 0,''))))result2

              fromjunion
              selectCodePattern,keyword1,keyword2,''keyword3,

              PATINDEX('%'+((SELECTSTUFF((
              SELECTpattern+'00'
              from#tblwherenameIndexin(1,2,3)
              orderbynameIndex
              FORXMLPATH(''),TYPE).value('.','NVARCHAR(MAX)'), 1, 0,'')))
              +'%',CodePattern)result1,-- index 1,2

              PATINDEX('%'+CodePattern+'%',((SELECTSTUFF((
              SELECTpattern+'00'
              from#tblwherenameIndexin(1,2,3)
              orderbynameIndex
              FORXMLPATH(''),TYPE).value('.','NVARCHAR(MAX)'), 1, 0,''))))result2
              frommunion
              selectCodePattern,keyword1,''keyword2,''keyword3,
              PATINDEX('%'+((SELECTSTUFF((
              SELECTpattern+'00'
              from#tblwherenameIndexin(1,2,3,4)
              orderbynameIndex
              FORXMLPATH(''),TYPE).value('.','NVARCHAR(MAX)'), 1,
0,'')))+'%',CodePattern)result1,-- -- index 1,2,3
              PATINDEX('%'+CodePattern+'%',((SELECTSTUFF((
              SELECTpattern+'00'
              from#tblwherenameIndexin(1,2,3,4)
              orderbynameIndex
              FORXMLPATH(''),TYPE).value('.','NVARCHAR(MAX)'), 1, 0,''))))result2
              fromt)

              --insert into
@tbl(CodePattern,keyword1,keyword2,keyword3,result1,result2)
              select*fromlwhereresult1>0 orresult2>0
```

```
                select*from@tbl


                if(selectcount(*)from@tbl)=0
                begin
                        update#tblseterr=
                        (SELECTSTUFF((
                        SELECT' '+CodeMasterName+','
                        fromdbo.SplitStringByWord(@text,2)kinnerjoin
                        [dbo].[CodeMaster]conc.CodeMasterCode=k.Result
                        wherec.CodeMasterCodenotin('03','00','04')
                        orderbyrefID
                        FORXMLPATH(''),TYPE).value('.','NVARCHAR(MAX)'), 1, 1,''))
                        wherenameIndex=@in
                end
                else
                begin
                        update#tblseterr=''wherenameIndex=@in
                end
        set@in+=1
end
declare@errvarchar(max)=''
selecttop 1 @err=isnull(err,'')from#tblwhereerr<>''
select@errerr


((SELECTSTUFF((
                SELECTpattern+'00'
                from#tblwherenameIndexin(1,2,3,4)
                orderbynameIndex
                FORXMLPATH(''),TYPE).value('.','NVARCHAR(MAX)'), 1, 0,'')))

select*from#tbl

droptable#tbl
```