



Faculty of Engineering and Information Technology
Master of Computing Program
Master Thesis

Enhancing Software Comments Readability

تحسين مستوى القراءة للتعليقات الخاصة بالشفرة الالكترونية

Author

Abed T. Othman

Supervisor

Dr. Derar Eleyan

This Thesis was submitted in partial fulfilment of the requirements for the
Master's Degree in Computing From the
Faculty of Engineering and Technology at Birzeit University, Palestine

Jan.2018

Enhancing Software Comments Readability

تحسين مستوى القراءة للتعليقات الخاصة بالشفرة الالكترونية

By:
Abed Othman

This thesis was prepared under the supervision of Dr. Derar Eleyan and has been approved by all members of examination committee:

Dr. Derar Eleyan
(Chairman of Committee)



Dr. Samer Zein
(Member)



Dr. Sobhi Ahmed
(Member)



Date of Defense: January -2018

Declaration of Authorship

Abed Othman declares that this thesis titled, 'Enhancing Software Comments Readability' and the work presented in it are my own. I confirm that:

This work was done wholly or mainly while in candidature for a master degree at Birzeit University.

Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.

Where I have consulted the published work of others, this is always clearly attributed.

Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work. I have acknowledged all main sources of help. Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date: *Jan. 2018*

سَمِ اللّٰهُ الرَّحْمٰنَ الرَّحِيْمَ

"قُلْ اِنْ صَلَاتِيْ وَنُسُكِيْ وَمَنْيَايَ وَمَمَاتِيْ لِلّٰهِ رَبِّ الْعَالَمِيْنَ، لَا شَرِيْكَ لَهُ

وَبِذَلِكَ اُمِرْتُ وَاَنَا اَوَّلُ الْمُسْلِمِيْنَ". صدق الله العظيم

(القرآن الكريم - سورة الأنعام)

Acknowledgements

First and foremost, I thank Allah for every things.

I would like to record my gratefulness to my supervisor, Dr. Derar Eleyan who has advised me and always helped to complete my thesis. Also for examiner Dr. Sobhi Ahmad and Dr. Samer Zain for a valuable comments and advices that helped me to publish this thesis.

I am extremely grateful to my parents, especially my mother, the candle that lit my way, and to my father “عليه رحمة الله” (may his soul rest in peace) who can never ever be thanked enough.

With my special, deep and warm thanks, and deep acknowledgment to my wife, who has endured a lot with me to make this work come to light. Her understanding, support, and looking after my children during my study all stand behind my success. Also I would like to thank my children Tahseen, Mohammed and Leen. My deep love and thanks are due to my brothers and sisters.

I would like to express my deep and sincere gratitude to Dr. Tawfiq Ammar of the Department of Languages and Translation at Birzeit University who helped me in reviewing the research questionnaire in addition to final report.

Finally, I would like to thank all my friends and colleagues who have given their support and help especially my best friends, Ali Jadda, Amr Miqdadi, Mohammad ZeinEddin, and computer center staff at Birzeit University.

Abstract

Programming comments are used to explain the code meaning and to understand communications between programmers or between programmers and editors (QAs, auditors, code reviewers). Because code with comments could be more descriptive, and easy to understand from code without comments and then easy to reuse or maintain. However, these comments are considered as part of the code without being tested or compiled, as they are written with high professional language or in natural language with spelling and structural faults, which can't be understood by others and this makes its existence superfluous.

The main aim of this research is to develop a tool that helps programmers to write readable comments on code and measures their readability according to predefined criteria. Also, this tool suggests comments and keywords to enhance the software readability by providing alternatives to both the keywords and the comment statements. These alternative terms are listed in local database in addition to online dictionary consumed from "Datamuse API that is a word-finding query engine for developers". On the other hand, the readability procedure will be measured by evaluating the comments readability from passing the comment text to function that used three different formulas Fog index, Flesch reading ease score, and Flesch-Kincaid grade level.

A questionnaire has been designed to compare readability between both the new comments from tool and original from human. This questionnaire has been disseminated to a target of (42 programmers and 35 students from computer science from BZU). From programmers the result was the comments from proposed tool have less complex words and took less time to read. But it did not significantly affect the understandability of the text; we guess that this comes from the high level of English of programmers and as [30] says that there a strong correlation between reading comprehension and vocabulary knowledge. So we can say that the tool reduced the complexity of the text and the time to read it. On the other hand, these two factors are important if we look to the cost effect of reduced time consumption to read text with code readable. The two factors above also influence any work related to code understanding. However, the result from students were the tool affected the understandability of text in addition to affecting the time of reading and text complexity were that the tool make new comments text more readable from changing the three studied variables in a positive way.

المخلص

تحسين مستوى القراءة للتعليقات الخاصة بالشفرة الالكترونية

التعليقات الخاصة بالشفرة البرمجية تستخدم لتوضيح معنى النص المصدري وتستخدم كوسيلة اتصال بين المبرمجين فيما بينهم والمحريين الآخرين كمرافقوا الجودة والمراجعون . لان التعليقات الخاصة بالشفرة البرمجية تكون أكثر توضيحا وسهولة في الفهم وتساعد في حالات إعادة الاستخدام والصيانة للنص المصدري الذي يكون بدون التعليقات الخاصة بالشفرة البرمجية . إن التعليقات الخاصة بالشفرة البرمجية تعد جزء من النص المصدري ولكن لا يتم فحصها أو ترجمتها إلى لغة الآلة. سواء كانت بلغة احترافية أو بلغة غير قوية او غير مفهومة وقواعد سيئة والتي سيكون من الصعب فهمها من الآخرين مما يجعل وجودها عبثيا. إن الهدف الأساسي من هذا البحث هو تطوير أداة تساعد المبرمجين لكتابة التعليقات الخاصة بالشفرة البرمجية أكثر سهولة من ناحية القراءة والفهم مع النص المصدري وقياس نسبة سهولة القراءة وفق معايير محددة. كذلك فان هذه الأداة ستقترح مجموعة من البدائل للكلمات الخاصة ببعض مكونات التعليقات الخاصة بالشفرة البرمجية بهدف تحسين مستوى سهولة قراءة النص المصدري ، المصطلحات البديلة ستسجل بقاعدة بيانات محلية بالإضافة إلى أخرى عبر الانترنت من خلال واجهة برمجية متخصصة باستعلام الكلمات من خلال محرك بحث للمطورين. من جهة أخرى فان الأداة تستخدم لقياس مستوى سهولة القراءة اقترانا يمرر التعليقات الخاصة بالشفرة البرمجية على ثلاث معادلات مختلفة هي

(formulas Fog index, Flesch reading ease score, Flesch-Kincaid grade level).

ليتم فحص الاداة صُممت استبانة لمقارنة سهولة القراءة بين كل من نص التعليقات الناتجة عن استخدام الأداة والاصيلة التي تم تجميعها من المبرمجين. وزعت هذه الاستبانة على (42 مبرمج و 35 طالب من قسم الحاسوب بجامعة بيرزيت) النتائج من المبرمجين كانت إن الأداة خفضت التعقيد في الكلمات وأخذت وقتا اقل للقراءة. لكن أثر نسبة فهم النصوص لم تتأثر لديهم. ويعزى ذلك لمستواهم العالي باللغة الانجليزية. لذلك يمكن القول إن الأداة خفضت من تعقيد النص وقللت من وقت القراءة، و هذان المؤشران مهمان إذا نظرنا للتكلفة التي سنقلها بتقليل الوقت اللازم لقراءة ال التعليقات الخاصة بالشفرة البرمجية مع النص المصدري. أيضا ستؤثر على أي دراسة ذات علاقة في فهم النص المصدري . إلا أن نتائج الطلبة ان الأداة أثر على نسبة الفهم للتعليقات الخاصة بالشفرة البرمجية كذلك وقت القراءة وتعقيد النص كانت تثبت وجود تغيير للأفضل.

Table of Contents

Acknowledgements.....	v
Abstract.....	vi
المخلص.....	vii
Abbreviations.....	x
1. Introduction	1
1.1. Motivation.....	2
1.2. Main objectives of the research.....	2
1.3. Contribution	2
2. Background.....	4
2.1 Software Code Quality	4
2.2 Code Review	7
2.3 Code Readability and Comments.....	8
2.4 More about Source Code Comment.....	11
2.4.1 Comments Type.....	13
2.5 The Case against Commenting.....	15
2.5.1 Programmer's Hubris	15
2.5.2 Laziness.....	15
2.5.3 The Deadline Problem	15
2.6 Conclusion.....	15
3. Related Work.....	16
3.1 Readability Formulas	17
3.1.1 Flesch Reading Ease	17
3.1.2 Flesch-Kincaid	18
3.1.3 Gunning-Fog Index Formula	18
3.2 Source Code Comments Assessment	19
3.3 Conclusion.....	21
4. System design and implementations.....	22
4.1 Methodology	22
4.2 System Design.....	23
4.2.1 Measurement Readability	24
4.2.2 Replacement Module.....	24

4.3	System Implementation.....	24
4.3.1	Implementation of Readability Measurement Module	24
4.3.2	Replacement Module	29
4.4	System Testing	32
4.5	Conclusion.....	33
5.	Experiment Design	34
5.1.	Design of the Questionnaire.....	34
5.2.	Research Hypothesis	35
5.2.1.	System hypothesis.....	36
5.3.	Survey Participants.....	36
5.4.	Conclusion.....	37
6.	Data analysis and Discussion	38
6.1.	Data analysis	38
6.2.	Discussion	45
6.3.	Conclusion.....	48
7.	Conclusion and Future Work.....	49
7.1.	Conclusion.....	49
7.2.	Suggestions for future work	50
	References.....	52
	Appendix.....	55
	Table of Figures:	55
	Table of Tables:.....	56
	Questionnaires data:	57
	Programmers Questionnaire first part:	57
	Programmers Questionnaire part 2.1:	60
	Programmers Questionnaire part 2.2:	62
	Questionnaire mapped table	64
	Students Questionnaire first part:	65
	Students Questionnaire part 2.1:.....	68
	Students Questionnaire part 2.2:.....	70

Abbreviations

CRS	Comments Readability system
IDE	Integrated Development Environment
SQ	Software Quality
CR	Code Readability
VS	Visual Studio
MSS	Microsoft SQL Server
BZU	Birzeit University

Chapter One

Introduction

“Good developers write good code; great ones also write good comments.”¹

(David Njoku)

Code writing when developing software life cycle is considered an important phase. This is because the next phases depend on this phase especially the software maintenance which needs to take the code maintainability, readability, and reusability into consideration. By this we should be careful about the code readability which is related to three terms mentioned before; also the readability becomes a key factor in software quality [1]. On one hand, the readability of source code with its documentation is grouped together to influence the software maintainability; on the other hand researchers noted that understanding code after or while developing is taking long time comparing with other software maintenance activities [1] [2]. That source code readability and documentation readability are both critical to the maintainability of a project. Other researchers have noted that the act of reading code is the most time-consuming component of all maintenance activities. One of the factors that affects this is comments which is defined as embedded documents and useful artifacts related to the code quality [2]. These comments existence is necessary for program comprehension especially the maintainers differ from programmers or main developers. For example, Mozilla source code involved about 15-20% as comments [2].

```
8 // Dear programmer:
9 // When I wrote this code, only god and
10 // I knew how it worked.
11 // Now, only god knows it!
12 //
13 // Therefore, if you are trying to optimize
14 // this routine and it fails (most surely),
15 // please increase this counter as a
16 // warning for the next person:
17 //
18 // total_hours_wasted_here = 254
19 //
20
```

*Figure 1-1 quote about importance of comments*²

¹ <https://www.red-gate.com/simple-talk/sql/oracle/how-to-make-comments-the-most-important-code-you-write/>
9-11-2017

² <https://www.pinterest.com/pin/383228249526156667/>

This research is concerned with creating a tool for enhancing source code readability, using predefined formulas to amend the source code quality by increasing the readability level of code comments. This tool will provide programmer suggestions to improve the readability level of written comments. The following sections will discuss the motivation behind this research, the contribution to the state of the art.

1.1. Motivation

The software development period including the maintainable process consumes over 70% of the software development time [3] [1]. This period is determined by the source code readability that if the code is difficult to understand then the maintenance process will take longer. The source itself is not enough to understand the purpose of written block, and programmer must use a documentation to describe this code as comments or inline-document. These comments are normally used as a guide showing what the programmer is thinking while writing every piece of code.

In software compilation and debugging process the comments are not considered or tested and ignored by the compiler, so the natural language is used without any limitation, in syntax, structure, words used, the length and the complexity of the comment statement. Most researches conducted in this field care about the code readability without considering the influence of comments in code readability. This motivates us to create a model to assess the comment and make enhancements on these comments.

1.2. Main objectives of the research

The main aim of this research is to develop a tool which helps developers to enhance the readability level of text for source code comments or any comments in code file. To achieve this aim, the following objectives should be achieved:

1. Critically analyze the readability existing tools, which measure the text readability.
2. Develop a tool based on the three formulas to measure software comments readability and propose suggestions to improve comments text readability.

1.3. Contribution

Develop a tool to assess the written comments by showing the readability level and provide the software developers with suggestions to change terms or words to improve the current comments

to enhance the level of text readability. Three main formulas are “Fog index, Flesch reading ease score, and Flesch-Kincaid grade level” will be compared to determine the readability level of traditional comments and with one which the research solution applied in. This will lead to more readable comments which improve the software quality and make the readability and understanding of code easier.

Chapter Two

Background

Software Quality which is assessed by many factors and there is more than one viewpoint about what is Software quality main factor and how we can measure it. In our research, we are concerned with the heart of these factors, which is the source code readability and focus on the comments written to describe and explain this source code. In this chapter, we discuss the code quality terms. Section 2.1 gives an overview of Software Code Quality; sec 2.2 discusses code review , then goes deeply into the Code Readability and its own Comments in sec 2.3; finally in sec 2.3, more about comments how to use it and when, in addition to comments types at the end of section.

2.1 Software Code Quality

Software quality (SQ) is an important factor of software life cycle. Developers always aspire and exert lots of efforts to develop software of high quality, free of faults and errors. Software Quality affects budget and time, which reflect on the plan of business risks and reduce the cost of software development life cycle [1] [2] [4] [5].

Software quality definition changes as the software development procedures and environments change. At the beginning, the focus was if system matches requirements, then how software was capable covering dynamic changes. In 1991, the International Organization for Standardization adopted ISO 9126 as the standard for evaluating software quality. This standard defines quality as the totality of features and characteristics of a product or service that bears on its ability to satisfy given needs” [15]. After ISO 9001 founded the SQ was used to measure the defects in software and check the degree of the systems to match user requirements, needed functions, and the system capability to obtain future requirement of changes [5] [6] [7] [8].

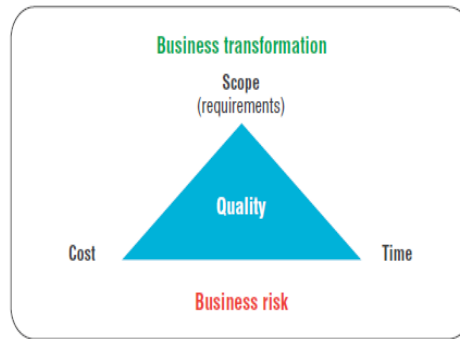


Figure 2-1 “The iron triangle” managing the balance between business transformation and risk [5]

The main aim of SQ is to consider the main three factors affecting software life cycle: user requirements, budget and time see (figure 1). Most of the software project budget and time consumed on operations, mainly, in fixing bugs or defected or missed requirements see (figure 2) [5].

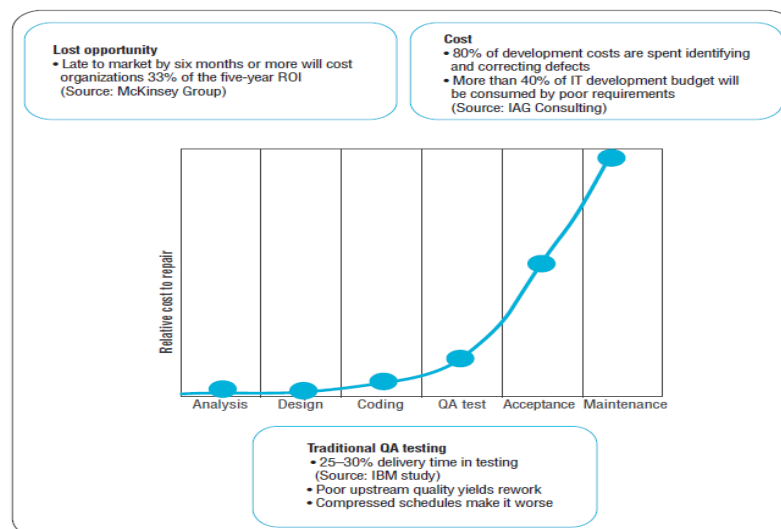


Figure 2-2: Opportunities to optimize quality and time to market exist throughout the application development lifecycle [5].

SQ assessed by number of factors or variables divided into internal and external categories of measurements; external criteria depends on user experience, while Internal criteria depends on background of software development especially about the code. The end user is not interested in internal criteria, which are not shown to him while this part is very important to programmer; end user is interested in external quality. Also, there are different criteria used to measure the SQ; one of them is a qualitative or subjective measure in with specific rules as security that can be measured as range not exact number and there are many factors affecting these measurements and could be different from one study to another. Another type is a quantitative or Objective which is measured as estimated and

the mix of the two measurements here the system functions may have more priority. To evaluate or measure SQ the following common quality attributes are used [4] [5] [6] [8] [9] :

- ✓ Availability
- ✓ Conceptual Integrity
- ✓ Interoperability
- ✓ Maintainability
- ✓ Manageability
- ✓ Performance
- ✓ Reliability
- ✓ Reusability
- ✓ Scalability
- ✓ Security
- ✓ Supportability
- ✓ Testability
- ✓ User Experience / Usability.

In this research we will focus in software maintainability, which is the ability of the system to fit the new changes easily in any phase of system life cycle. While the new changes may be affected by the new services, system workflow, main features, and system interface. Software readability or “code readability” is an important part to assess software maintainability: “How quickly can a new developer understand the code in shortest time” which is the main point of our study [4] [5] [10] [11] [9] [12].

Development period of software is short if compared with software life cycle while code maintenance and enhancements consume over 70% of the total life cycle and cost of a software product [3]. This phase becomes more and more difficult without comments or text documentation about the program, which is the term we care about in our research. In most times the person who updated or upgraded the system differs from the coder i.e. there is huge code of open source. The probability of software development bugs occurrence becomes more and more if there are no communications tools among programmers to explain their code; from this point the comments is a necessary factor that helps and supports programmers to understand the code especially if the software is used in critical field [1] [3].

2.2 Code Review

A Code review is considered an important stage, role or function in modern software development [13]; this process is done to identify bugs or errors in source code and improve code quality [14] [15]; this will help in optimizing the software quality which is a function of the degree of reviews made during the early life of a software development process [16]. In addition to helping developer to improve their own skills, thereby mitigating the occurrence of errors in the later stage of software development process; moreover, as we mention before the errors detected during the early life cycle of software are least expensive to correct [16]

Code review provide number of advantages in addition to list that we said before [15]

- *It improves team communication.* Code review gets people talking, collaborating, and building trust in addition.
- *Code review provides a safe opportunity to mentor and teach junior team members.* Newer developer can work without thinking about critical faults because there are seniors who will be in back of them. Also, these seniors can be sure about the code quality.
- *It propagates code expertise.* “Code review increases your whole team’s knowledge about the entire code base. The alternative – having only one or two team members who are experts on all (or even parts) of your code – is a huge risk” [29].

Reviewing code needs techniques and methods to evaluate development team and cover all small and critical faults some of this methods are [15]

- ***Over the Shoulder***

In this method, there is reviewer standing behind the developer and this developer reviews code with his partner step by step, from this methods bugs corrected immediately. This informal method works well when developers are in the same office and can schedule time to work together.

- ***Meeting-Based Walk-Through***

This method is like Over the Shoulder reviews, except that teams get together and review code on a projector, using a virtual meeting, or with code printed out.

- ***Email Pass-Around***

In this method team members spread code file by email other will review and give his feedback by email with all of comments also. This can be used when team members distributed in different geographically area.

- ***Tool-Supported***

This is considered as fastest and most efficient method, tool-supported codes reviews use dedicated software to automate uploads and facilitate online discussions. This tools should be include the following functions: [15]

- Automated File-Gathering
- Combined Display: Differences, Comments, Defects
- Automated Metrics Collection
- Workflow Enforcement
- Clients and Integration

Finally reviewing of code may not be easy if there are no comments on code, so the existence of comments is very important to reviewer In addition to the quality and readability of these comments. If these comments are not readable the reviewer will ignore it and take it will more time to understand the code and may jump over critical errors. This can provide evidence for our research to be used good and efficient comments.

2.3 Code Readability and Comments

Most companies' developers are working as team instead of working individually, thus most of software code written by different teams; this defiance may in human resource or in physical area, even the developer joins or leaves the team; the code must be reusable and maintainable, so it is important that this code must be understandable. By this the readability becomes priority, also readability has always been the reason behind maintainable code, so readability is needed. Thus readability becomes a dominant topic in software engineering especially in software quality [3].

Code readability is very important in software development process. There are many definitions for this term; it is defined as human judgment of how easy a source code is to understand by programmers. Or as the process by which a programmer makes sense of programming code. And as the difficulty of understanding the functionality of modules or software component. There is close relation between the source code readability and its maintainability. Also, readability is an essential determining characteristic of code quality [8] [17] [18] [19] [20] [10].

Readability is one important quality attribute for software source codes. And it is key factor in overall software quality. "Maintenance programmers' tasks are analogous to those of archeologists

who investigate some situation, trying to understand what they looking at and how it all fits together. To do this, they must be careful to preserve the artifacts they find and respect and understand the cultural forces that produced them” [10]. The difficulty of understanding code influences the high cost of software maintenance. Moreover, reduce reusability of this source code especially a source code may be considered a readable one for a programmer but it might not be for another. This is applied in two situations when others need to understand the code or the author of the code after a while (month, year) he could not remember the written code or took a long time to do. Most researches show that the reading code and understanding it take longer time compared to maintenance activities. So we can find many companies that care about code readability especially when talking about commercial software and the time as we mention before is one of the main factors that we should take into consideration during the development cycle process. As authors in [4] mention there is model suggesting that “one phase of software inspection should be a check of the source code for readability to ensure maintainability, portability, and reusability of the code. Proposed adding a dedicated readability and documentation group to the development team”. There are many factors that determine the readability of software such as simplicity of control sequences, comments, and top down approach and so on. Also authors in [20] Determines of code readability is not simple as text readability measure, that because the structure of the code. It comes with how the readers’ understanding of the semantic relations between propositional elements of the code [18] [19] [10].

“Unfortunately, computer system documentation is difficult to understand, badly written, out-of-date or incomplete”. [21]

The complexity of code and readability are not the same although they are closely interrelated; complexity is code property based on the problem domain, The way how to solve each scenario also will not be avoided completely in developing process. “Whereas readability is an accidental property that can be avoided independent of the problem domain or problem complexity” [18]. Code readability is not easy to measure by a deterministic function same as maintainability, reliability, and reusability. Readability measures formula care about and changed by independent individual elements, such as identifiers, statements, comments, indentation, and program style. But complexity measurements depending on static components as well dynamic interactions among program components. [18] [22]

The structure of code has impact on readability level. For example, dividing functions to sub function helps programmers to follow the program process, and make debugging mode easier. Besides, this function should have clear name with meaning that gives the reader hint about what the function does [9]. In addition to the codes style and structure, its readability is affected by major factors that make code more readable and give short description about the code which is code comment.

Using comment reduces the time that is used to explain code to other colleagues, most of junior programmers look for comment to understand the code. But at the same time if the code included unclear comments, they cause embarrassment for the programmer to understand the code and the existence of these comments is inefficient. We define code readability term as efforts which by the reader to realize the written code from its comments and what is the degree of it i.e. easy, hard...etc.

The main affected side of software quality is readability as discussed before, if code is understood by others so the testing, maintenance phase will go smoothly and less cost when there is no hint or document text, which can be as internal or external text see the figure bellow.



Figure 2-3 Code Should Be Easy to Understand [23].

Although most of evidences show that there is importance of commenting the code, there is programmers adopts the opposite point of this view. They say the good code not need to commented it, because the code itself is clear and understandable [8], Robert Stoll, [24]says that “The frequency of comments sometimes reflects poor quality of code. When you feel compelled to add a comment, consider rewriting the code to make it clearer”. developers don't need to commented every line of code, because when programmer need to comment every line of code to make it understandable, this might indicate that the code is low quality or have lack in structure, In this case programmers should rewrite it instead of commented it. However, as we mentioned before comments is necessary in development process and maintenance also to make code more readable even when the code

itself quality is low. For this point we focus in the comments quality and readability that will use in system documentation later [17] [23] [4] [18] [3].



Figure 2-4 is comments existence necessary after a while [23].

comments readability means a detailed information and description about source code see figure 4, which will be shared between software engineering team with different tasks and professionalism level, to hold proper understanding for the system functions or properties, It plays a key role especially in large systems [18] [3].

Students study codes from examples in textbooks, outline documents, other colleague's code; these resources and others are used as learning tools in academic environments. These codes or programs can be unhelpful if their readability is too high or if it is a low level one for these students. That makes understanding the code hard and increases the difficulty of maintainability or reusability of these codes [22].

2.4 More about Source Code Comment

The Software code divided into two types: code itself, and comments on code. Comments are a blocks of source that the compiler ignores in compilation process, so you can put in it any information that you want. However, programmer should keep in mind that the main aim of using these comments are notation the source code and help the developer and other software implementer team members to understand the source code in processing of upgrading or bugs fixing or to make software fits with new or modified requirements, in other words give aid in software maintenance and therefore reduce maintenance costs. Several researchers have conducted experiments showing that commented code is

easier to understand than code without comments. Also these comments are used as internal documentation for code and system [25] [8] [17].

Importance of comments becomes more and more when we talk about developing complex applications or systems. Because there are many developers who may be distributed in many geo areas with different level of experience. So the comment here becomes the communication tunnel between them to understand what others mean in specific block of code. But these comments must be suffice why they used for.

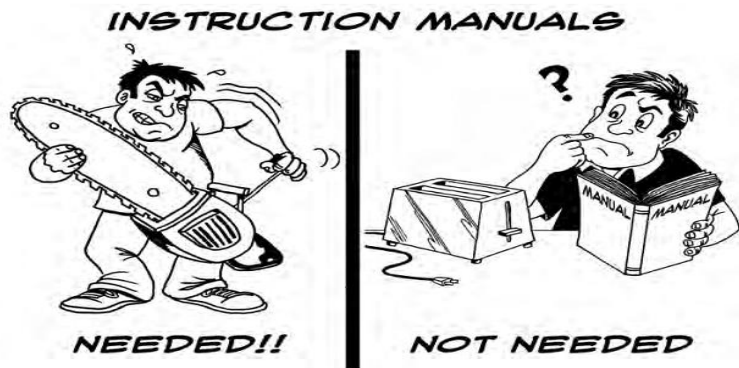


Figure 2-5 when comment is needs [23].

We should keep in mind that comments used to make code more readable not to replace it. Also if code is bad we should make changes to have good code not have good comment. However, we can say that bad comments are worse than no comments at all, for this “We should only be writing comments when they really add something “ [25]. By the way, there are some concerns which should be taken into consideration when writing comments as [25] [9] mentions:

- ✓ Quality of the comments is more important than quantity.
- ✓ Good comments explain why, not how.
- ✓ Code should not be duplicated in the commenting.
- ✓ comments should be clear and concise
- ✓ functions should start with a block commenting
- ✓ comments should be indented the same way as the rest of code
- ✓ Comments should be used in a way that reads naturally before a function, not below.

There are different styles used in commenting, each language almost has its own commenting style Most of which are set in green color, i.e. C language comments come in blocks between `/*` and `*/` and

can span any number of lines. C++, C#, Java add the single line comment that follows //, and vb.net used single coma ‘ [25].

2.4.1 Comments Type

The using of comment is founded in the main three categories as [12]categorize them:

- Documentary comments
- Functional comments
- Explanatory comments

2.4.1.1 Documentary Comments

This type focuses on the history of file system and main properties of this file like author, date, updated date, copyright, introduction of class or source file as the following example of php comment in file header.

Example of comment in file.

```
/**
 * Date sep. 2015
 * Academic Business Unit Colleges Entity class. (DBA_OBJECTS.VIEW)
 *
 * @package Entity
 * @subpackage teaching
 * @link https://ritaj.birzeit.edu/
 * @author Abed T. Othman <aothman@birzeit.edu>
 *
 * @orm\Entity
 * @orm\Table(name="asm.bu_collages")
 */
```

2.4.1.2 Functional comments

This type of comments is used to serve single process in the development cycle. This famous example used here is “to do”. Also there are three positions we can find it [12] :

- Bug description
- Notes to co-developers

- Performance/feature improvement possibilities

2.4.1.3 Explanatory Comments

This type of comment is most commonly used one. Its function comes tied with its name to give more details about the item commented. However, there are many items which have this type of comment [24] as follow:

- Startup code
- Exit code
- Subroutines and functions
- Long or complicated loops
- Weird logic
- Regular expressions

Example of using comment in function and regular expression [12]

```
void DumpHrefsClean(String inputString) //same as above, commented
{
    Regex r;
    Match m;
    r = new          Regex("href      #This looks for the string 'href'
    \s*=\s           #followed by white spaces, '=', ws
    (?              \"(?<1>[^\"]*)\"    #a ':', + a group in \"\", no \"\" in it
    |                #or
    (?<              1>\s+))\",      #a group followed by non-spaces
    RegexOptions.Ignore WhiteSpace|RegexOptions.IgnoreCase|
    RegexOptions.Compiled);
    for (m = r.Match(inputString); m.Success; m = m.NextMatch())
    {
        Console.WriteLine("Found href " + m.Groups[12] + " at "
        + m.Groups[12].Index);
    }
}
```


2.5 The Case against Commenting

2.5.1 Programmer's Hubris

There is a number of programmers who see that there is nothing hard to do, everything is possible and easy to understand and will be developed in a smooth way. But in reality this is not 100% correct; many programmers can't remember their code after a period of a year for example. "Truth is, most of the time it will take a lot of time and effort to understand undocumented code, even if the code has not been obfuscated intentionally" [12].

2.5.2 Laziness

"All time saved by not commenting during the coding process is made up more than twice at least by inserting comments and providing documentation after the coding process is finished"[24]. Because nowadays there are softwares used to generate documentation by automatic collection of comment, the writing comment will save the time of writing documentation after software development is completed [12]

2.5.3 The Deadline Problem

The project deadline will not be the problem of omitting comments, because the time taken to write comments that the developer wishes to save while he is in development phase will be less than that one needed to fix bug with uncommented code after a while. [12] [26]

2.6 Conclusion

Software quality depends on many factors to be measured, some of them could be quantifying and others qualifying. Also there are many parts of software affected by these measurements, in this research we emphasize code part and its related factors that could affect its characteristics to be set as good or bad. This creates challenges for developer's aspirations and exerts lots of efforts to develop software of high quality, free of faults and errors.

Code review is considered as good stage in software development lifecycle. This reviewing could be affected by outline and helping part which is comments, this part which is not passed to compiler to check it or assess it, for this there are many studies done on this area to support the idea from existence of these comments, and how much these comments are useful for programmers and others.

Chapter Three

Related Work

There are many factors used to defined text complexity; some of these factors are quantitative measures, which place emphasis on the characteristics of the words themselves that will be used to measure complexity of sentences contains them. Also quantitative text measures are just concern with the words' properties that could be calculated and measured. These measurement functions are defined as readability formulas that measure semantic difficulty and sentence complexity. To create readability formula, there are five factors to be taken into account to measure the complexity [26]:

1. Average sentence length
2. Number of different hard words
3. Number of personal pronouns
4. Percentage of unique words
5. Number of prepositional phrases

The first step of getting readability level of text is at the word level. The length of word gives indicator about the degree of understating it. On the other hand, the number of syllables that the word contains gives another indicator about word level complexity, for the words with single-syllable in most of the cases are considered easier to understand than words with multi-syllable. Also the words usually used in general or word frequency used are assumed to be more familiar to the reader (i.e. dangerous, education) [26].

Also there are many researches and metrics performed to check and assess the text readability. What are the factors that affect the readability of text and how are these factors used to measure text reading difficulty? To make readability level suitable for target part of audience, most of the metrics and researchers use formulas that grade the level as level of education, year that the people should hold to understand the assessed text. However, the same way is almost done by the coder readability, how to make code readable for programmers who write the code and others who will update or reuse this code. The main factor of making code readable is comment that describes this code. In section 3.1 will discuss three formulas that are used to measure text complexity, In section 3.2 we will show what others do in this field and make a comparison between our work and others' work.

3.1 Readability Formulas

There is many readability metrics or formulas used to analyze source code comments. In our research we used the following formulas.

3.1.1 Flesch Reading Ease

This formula designed by Rudolph to measures the difficulty of text document context, also used as indicator or grading a difficulty of understanding reading content in English. This grading depends on several factors that affect the text content, such as: word length, sentence length, word form, and syllables or letters. For this measuring tools formula the result was the text with shorter sentences and words is the more readable. The grading coming as the following table shows that high score indicates a document that is easier to read. Lower scores indicate a document that is more difficult to read as table 1 shows [18] [11] [27].

Reading Ease Score	Description	Predicted Reading Grade	Estimated Percentage of U.S. Adults
0-30	very difficult	college graduate	4.5%
30-40	difficult	college grade	33%
50-60	fairly difficult	10 th -12 th grade	54%
60-70	standard	8 th -9 th grade	83%
70-80	fairly easy	7 th grade	88%
80-90	easy	6 th grade	91%
90-100	very easy	5 th grade	93%

Table 3-1 flesch reading ease score to assess the ease of readability in a document [27]

The following is the algorithm to determine the Flesch Reading Ease [18] [11] [27].

- ✓ Calculate the average number of words you use per sentence.
- ✓ Calculate the average number of syllables per word.
- ✓ Multiply the average numbers of syllables per word multiplied by 84.6 and subtract it from the average number of words multiplied by 1.015.
- ✓ Subtract the result from 206.835.

Algorithm: $206.835 - (1.015 * \text{average_words_sentence}) - (84.6 * \text{verage_syllables_word})$

3.1.2 Flesch-Kincaid

It is another formula of text readability measurement designed by Rudolph Flesch use the same core measures (word length and sentence length) as Flesch Reading Ease but it uses different weighting factors. The following is the algorithm to determine the Flesch-Kincaid grade level [27].

- ✓ Calculate the average number of words you use per sentence.
- ✓ Calculate the average number of syllables per word.
- ✓ Multiply the average number of words by 0.39 and add it to the average number of syllables per word multiplied by 11.8.
- ✓ Subtract 15.50 from the result.

Algorithm: $(0.39 * \text{average_words_sentence}) + (11.8 * \text{average_syllables_word}) - 15.9$

In the Flesch reading-ease test, higher scores indicate material that is easier to read; lower numbers mark passages that are more difficult to read. Scores can be interpreted as shown in the table below [1].

Score	Notes
90.0–100.0	easily understood by an average 11-year-old student
60.0–70.0	easily understood by 13- to 15-year-old students
0.0–30.0	best understood by university graduates

Table 3-2 Formula of Flesch-Kincaid [4].

3.1.3 Gunning-Fog Index Formula

Is readability measure formula used to measure English text readability, and the result is score, which gives an indication about the formal education year that the person needed to be able to understand the text from its first reading [5] for more information about level see table 2.

	Fog-Index	Estimated Reading Grades
	17	College graduate
	16	College senior
	15	College junior
	14	College sophomore
Danger line	13	College freshman
	12	High school senior
	11	High school junior
	10	High school sophomore
Easy Reading Range	9	High school freshman
	8	Eighth grade
	7	Seventh grade
	6	Sixth grade

Table 3-3 gunning's fog-index level [27]

The following is the algorithm to determine the Gunning-Fog index.

- ✓ Calculate the average number of words used per sentence.
- ✓ Calculate the percentage of difficult words in the sample (words with three or more syllables).
- ✓ Add the totals together, and multiply the sum by 0.4.
- ✓ Algorithm: $(\text{average_words_sentence} + \text{number_words_three_syllables_plus}) * 0.4$

3.2 Source Code Comments Assessment

In coding area there are many tools that used previous formulas or create their own formula. An approach for quality analysis and assessment of code comments by [8], Steidl et al. The provided approach defined a model based on categories of the comments. Researchers applied a machine learning technique on developed application which is programed in Java and C/C++. Authors used a metric to evaluate coherence between codes and comment of methods for example, the name of routine and its related comment. Also they used another metric that investigated the length of experimented comments. As for coherence, authors compare the words in comments with ones that founded in the method name. And the aim of using length of comment is coming from assumption that the short inline comment may contain less information compared with long ones. To apply this study they use survey that distributed over 16 experienced software developers. This work related to our work that by assessing the source code comment, but authors do not care about the readability level of comments as written text; they care if there is a relation between the code and the comment itself but in our research we focus on readability level of comments and its words completion to achieve the purpose of existence of these comments.

Authors of [2], Aman et al. in their research collected methods for java programs from six popular open source applications. They apply analyses on comments from collected datasets; to do this they conducted two preliminary studies on words appearing in comments and on amounts of comments. The result was most comments are categories as short sentences that contain at most 10 words. And the methods that inner code has more line of comments could need more changes in feature. Therefore, it would require more time to fixes especially after the product set as production version. Finally they go as the good comment may write to cover the poor code. This result may conflict with our work that we

can use good comments beside good source code. Also if these comments are not as user expected, we can improve the readability to more useful without affecting the code quality itself.

According to [17] Khamis et al. They develop Javadoc Miner tool to assess the quality of one type of comments which is in-line documentation by using a set of heuristics. To assess the quality of language and consistency between source code and its related comments. Authors measure the readability of comments by assessing the quality of language that comments were wrote with heuristics by counting the number of tokens, nouns, and verbs, calculating the average number of words, or counting the number of abbreviations. Also they used Fog index or the Flesch reading ease level to assess the readability level of comment text. The main aim of authors in this research is to detecting inconsistencies between code and comments, by checking the all properties of methods and even this properties documented in comments and explained as others can understand it. Authors found that the comments are not up to date which causes misunderstanding in working of these methods, Also authors notice that the codes which are well commented have less fault or problem reported than ones which have bad comment that have more fault and bugs.

This research aims to define the factors affecting the quality of source code comments and one of these factors is comments readability which we care about in our research, but they go to specific type of comments and its related quality and were very basic and not give alternative comments to get new readability level.

Researchers in [28] Were Created two data sets from tow corpora which were Penn Discourse Treebank and the Simple English Wikipedia corpora to be used as sample in there research and apply the researched feature that used to assess the complexity of text. These feature were divvied to 5 groups as surface, lexical, syntactic, cohesion and coherence features. They found that coherence features is needed to be in combination with others and if this features dropped from combination there is significant decrease in accuracy, this led to result as there is a strong correlation between text coherence and text complexity.

Researchers of [29] amid to prove that relationship between the fault-proneness and the commenting manner in methods declared in Java classes. They focus on two types of comments which were documentation comments and inner comments.to achieves their aim they used two methods (Analysis-1 and Analysis-2). The result of this research was a functions with inner comments is more faulty than a non-commented method, also using of comment may be give indication that programmer write poor code or faulty code.

3.3 Conclusion

Many researches and metrics performed experiments to check and assess the text readability. Also there are formulas used to assess the readability level of required texts, and change levels to make readability level suitable for target part of audience, most of them grading the level as number of education year that the people should hold to understand the assessed text. Most of studies took care of code itself and how to be more readable; some of them took part the comment effect of how these comments make code more readable.

Chapter Four

System design and implementations

In this chapter we will discuss the Implementation of system Comment Readability tool (CRS). The main objective of using this tool is checking the readability level and giving alternative words or terms to specific complex or unreadable words. In section 4.1 will discusses methodology used while developing the tool and the algorithms used to check the readability and replace with alternatives. In sec 4.2 we discuss the system design with main modules, in sec 4.3 system implementation phase deeply discussed, and finally the system testing in sec 4.4.

4.1 Methodology

The main goal of our research is make enhancement in code comments readability, therefore to make understanding code and related works processes (maintainability, reusability, and reviewing) easier and this will achieves the purpose of the existence of the comments. We go into building CRS “Comments Readability system” as a tool to be used by programmers to verify the readability of their own comments in development phase, and suggest improvements to enhance the comments readability level to be more understandable and valuable to anyone who will be in process of using this code. This tool will use three formulas to check the current comment complexity level, by this immediate evaluation new terms or words will be suggested to programmer, The Flesch Reading and gunning fog formulas with modification of their measuring parameters beside database of suggested complex word will used to assess the writing comment. Then mark each word not satisfying the preconfigured criteria and score each comment statement. This will provide opportunity to the programmer to go through each marked word and provide readable alternatives from the database or corpus which will be directly connected to the tool.

4.2 System Design

The proposed system used consists of two modules, the measurement readability module, and the replacement words module. As following figure 1 shows, the general processes that is executed in each phase of comment text readability measurement and alternative terms suggestion.

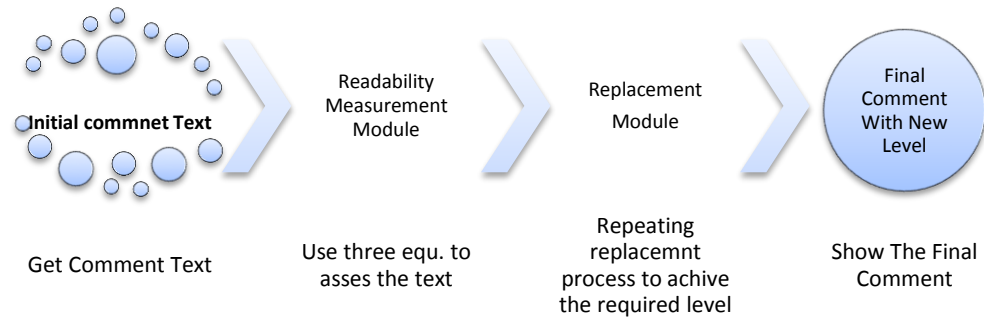


Figure 4-1 system general steps

In each part of system there is sub process executed to get at the final stage the required text with simple words and terms the following diagram (4.2) the overall system process shows how text passes through system modules and what is happening in each stage .

Text Readability enhancement Process

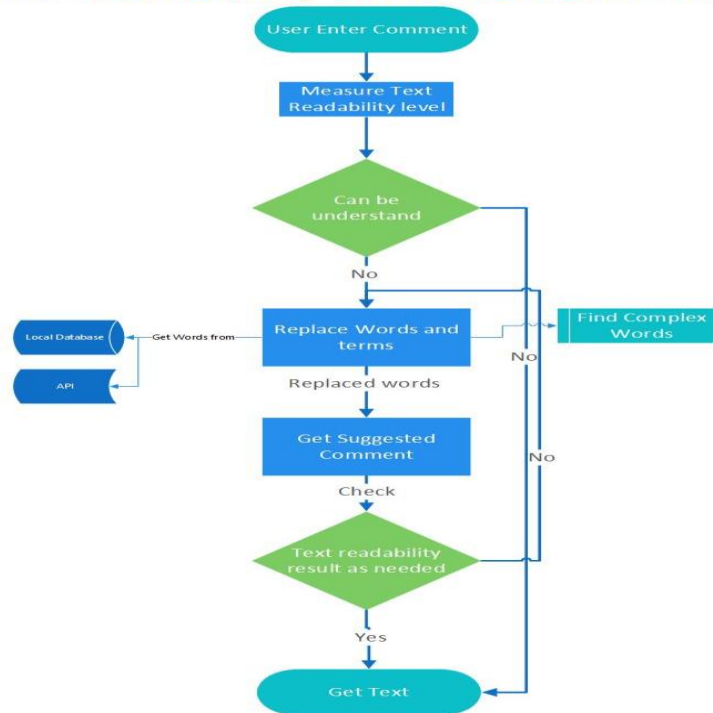


Figure 4-2 overall system process

The main two modules that system consists of are Measurement Readability and Replacement Module. The following two sections explain what is happening at each module stage.

4.2.1 Measurement Readability

This module is used to measure the comment text readability level by using three equations from three formulas used in this research. To assess the readability level for a text of target comment and extend the complex words from the text and set those as recommended words that should be replaced.

4.2.2 Replacement Module

This module is used to replace the suggested word from local database or from online database the consumed from API. The replacement term retrieved from local replaced automatically but from online it gives to the user a list of suggestions selected manually and which term is more readable form him/her. On the other hand, the listed term is scored by API teams to show the most suitable term for requested word as semantic or as generally used in daily life. By combining these two ways we will get more options for current text to determine the best alternative word, which gives other people the chance to understand what the writer means from these comments.

4.3 System Implementation

4.3.1 Implementation of Readability Measurement Module

This module is considered the heart of the proposed tool. It depends on three formulas to measure the level of text readability (fog index, Flesch reading-ease, Flesch-Kincaid). The following figure shows the internal process done from entering text, check readability from three formulas as functions. Finally get the level or readability as number with listing of complex words that could be replaced with simple ones.

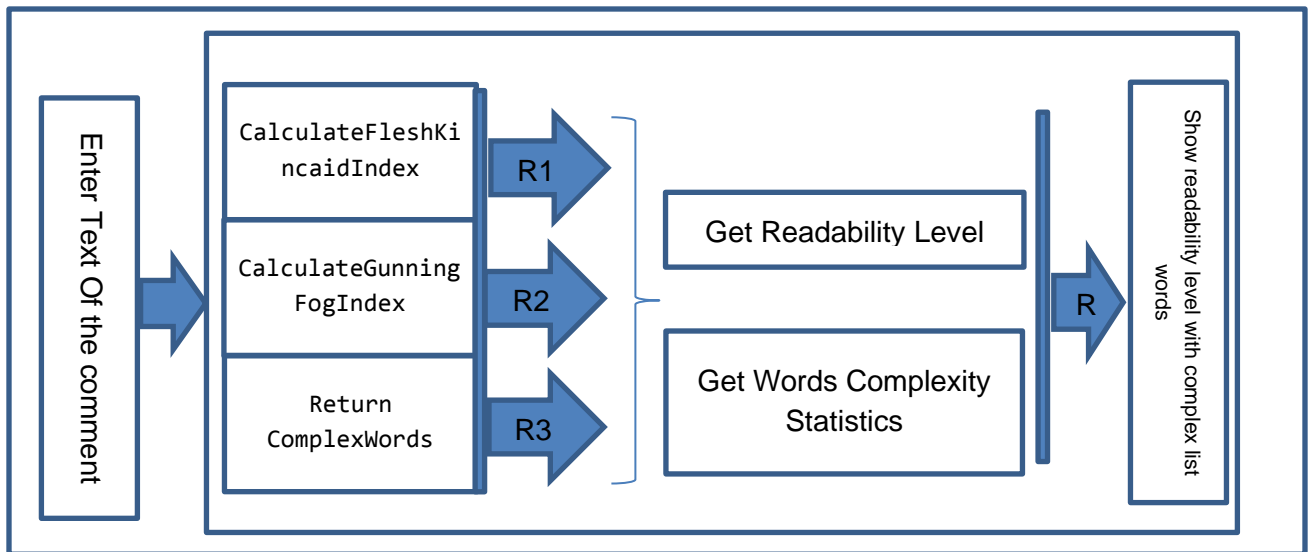


Figure 4-3 Module Functions and process

Each calculation formula is created as individual function called from main screen, in addition to supported functions used to extract entrances and extract words from each sentence.

The Following Code for Fog Index:

```
''' <returns>Gunning-Fog Index</returns>
```

```
Public Shared Function CalculateGunningFogIndex(inputstring As String) As Double
```

```
Dim sentencecount As Integer = BasicNLP.SegmentSentences(inputstring).Length
```

```
Dim tokens As String() = BasicNLP.Tokenise(inputstring)
```

```
Dim complexwords As Integer = BasicNLP.CountComplexWords(tokens)
```

```
Dim wordcount As Integer = tokens.Length
```

```
Dim indexval As Double = 0.4 * ((Cdbl(wordcount) / sentencecount) + 100 * (Cdbl(complexwords) / wordcount))
```

```
Return indexval
```

```
End Function
```

As we see the formula depends on word count, the count of sentences, complex word (this part depends on the count of Syllables and use as 3 and more), and the word count in submitted text. The score as we mentioned before, while it increases the readability also increased. The following figure shows the processes as entity working together to produce the indexing value of current system readability level.

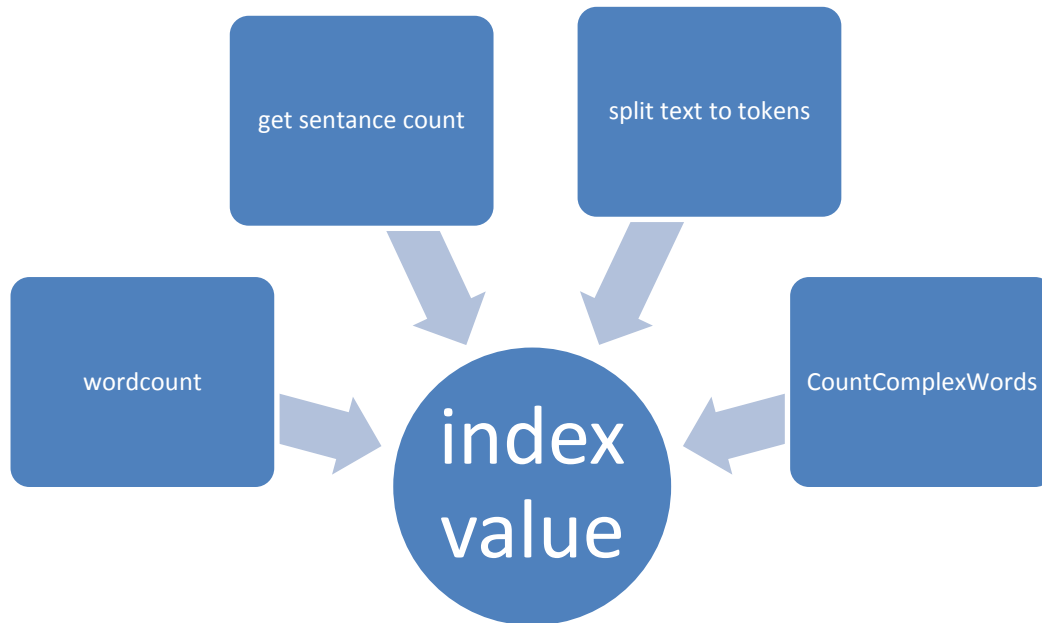


Figure 4-4 Figure of main function of getting indexval from Fog Index equation is in “sec 3.1”

Calculate FleshK incaid Index

```

Public Shared Function CalculateFleshKincaidIndex(inputstring As String) As Double
    Dim sentencecount As Integer = BasicNLP.SegmentSentences(inputstring).Length
    Dim tokens As String() = BasicNLP.Tokenise(inputstring)
    Dim syllablescount As Integer = BasicNLP.SyllableCount(tokens)
    Dim wordcount As Integer = tokens.Length
    Dim indexval As Double = 0.39 * ((Cdbl(wordcount) / sentencecount) + 11.8 * (Cdbl(syllablescount) / wordcount) - 15.59)
    Return indexval
End Function
  
```

This formula depends on different parameters which fog depends on which is syllables count.

And the following function is used to get the complex words, syllable count, and return the complex word into array of words to be changed after the whole function and process is done for this phase of module.

```

Public Shared Function SyllableCount(word As String) As Integer
    word = word.ToLower().Trim()
    Dim count As Integer = System.Text.RegularExpressions.Regex.Matches(word, "[aeiouy]+").Count
    If (word.EndsWith("e") OrElse (word.EndsWith("es") OrElse word.EndsWith("ed"))) AndAlso Not word.EndsWith("le")
    Then
        count -= 1
    End If
    Return count
End Function
  
```

```

Public Shared Function SyllableCount(tokens As String()) As Integer
    Dim count As Integer = 0
    For Each token As String In tokens
        count += SyllableCount(token)
    Next
  
```

```

Next
Return count
End Function
Public Shared Function CountComplexWords(tokens As String(), Optional syllablecountconsideredcomplex As Integer =
3) As Integer
    Dim count As Integer = 0

    For Each token As String In tokens
        If SyllableCount(token) >= syllablecountconsideredcomplex Then
            count += 1
        End If
    Next
    Return count
End Function

Public Shared Function ReturnComplexWords(tokens As String(), Optional syllablecountconsideredcomplex As Integer =
3) As ArrayList

    Dim complextokens As New ArrayList

    For Each token As String In tokens
        If SyllableCount(token) >= syllablecountconsideredcomplex Then
            complextokens.Add(token)
        End If
    Next

    Return complextokens
End Function

```

By measurement process the result of text ratibility can be evaluated and determined the complex words that should be replaced to make the text more readable therefore more understandable.

The following Figure shows the tool main screen, the screen has many parameters with two windows: first one is “Text body” The text which is written by programmer “Comment” context, each complex word will be highlighted to give indication that the written word is classified as complex and draws programmer’s attention to change it. The second window is “Suggested Text” that gives suggestion to replace some of the words found to be complex or if there is a simple alternative for it from local database. The figure has many labels that display the result of measuring readability for formula used to measure text readability. Also there is list of word complex and in front of it there is list of words that have been changed in suggested text.

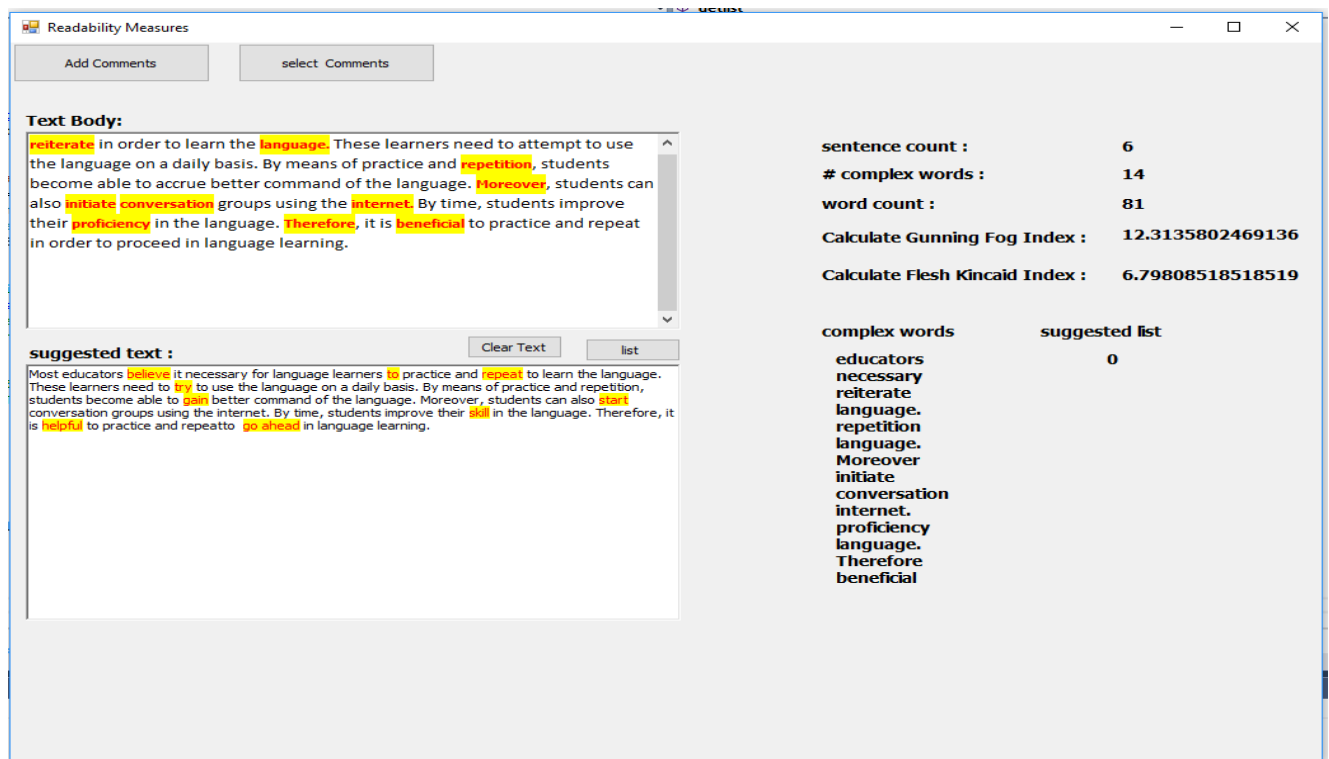


Figure 4-5 main Screen of system

The following figure show how will the readability level displayed to end user.

sentence count :	6
# complex words :	14
word count :	81
Calculate Gunning Fog Index :	12.3135802469136
Calculate Flesh Kincaid Index :	6.79808518518519

Figure 4-6 system calculations section view

The following is for listed the complex words that should be replaced as it's measure as complex words and the number of syllable more than the threshold values.

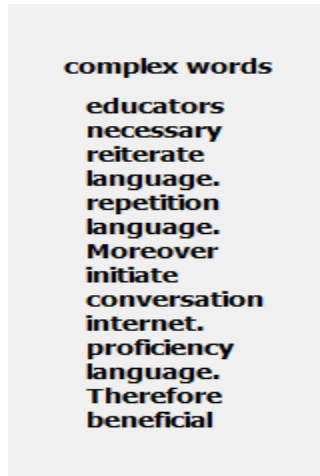


Figure 4-7 system complex words list view

4.3.2 Replacement Module

In this phase of system implementation we care about the words used to be substitution for original ones. And make sure the term is not complex and usually used. For this reason, we used local and online database for replacement. In this section we place emphasis on main functions used to get best result and be sure the enhancing of readability level is as we imagine.

4.3.2.1 Local Database

The local database consists of a number of complex words with simple alternative words collected from internet web sites. The following figure (8). Shows sample of it.

	Prime_Word	Alternative
1	accrue	gain
2	adjacent to	next to
3	advantageous	helpful
4	allocate	divide
5	apparent	clear
6	ascertain	learn
7	attempted	tried
8	beneficial	helpful
9	by means of	by
10	capability	ability
11	category	group
12	close proximity	near
13	commence	start
14	comply with	follow
15	component	part
16	concur	agree
17	consolidate	combine
18	constitutes	forms
19	convene	meet

Figure 4-8 sample complex words with simple alternative

The functions of changing words loop through the text of comment and replace all words found with alternative without user intervention.

```
For Each row As DataRow In ds.Tables(0).Rows
    If st.Contains(row.Item(0)) Then
        ReplaceAll(rtb_suggested, row.Item(0), row.Item(1))
    End If
Next
```

Figure 4-9 code for Replacing terms from local database

Where DS is dataset with all terms from local database and “rtb_suggested” is the control which contains the text. After that the first result shows as the following figure (10).

Text Body:

These learners need to attempt to use the language on a daily basis. By means of practice and repetition, students become able to accrue better command of the language. Moreover, students can also initiate conversation groups using the internet. By time, students improve their proficiency in the language. Therefore, it is beneficial to practice and repeat in order to proceed in language learning.

suggested text :

These learners need to try to use the language on a daily basis. By means of practice and repetition, students become able to gain better command of the language. Moreover, students can also start conversation groups using the internet. By time, students improve their skill in the language. Therefore, it is helpful to practice and repeat to go ahead in language learning.

Figure 4-10 example of using local database for replacement the terms.

Each word replaced from original text is highlighted as yellow with changing the color to red in both text boxes (text body, suggested text).

4.3.2.2 API Source

This is another source of word replacement; this source returns list of suggested words from API service consumed by sending the word that we need to replace or check whether this word has an alternative simple term.

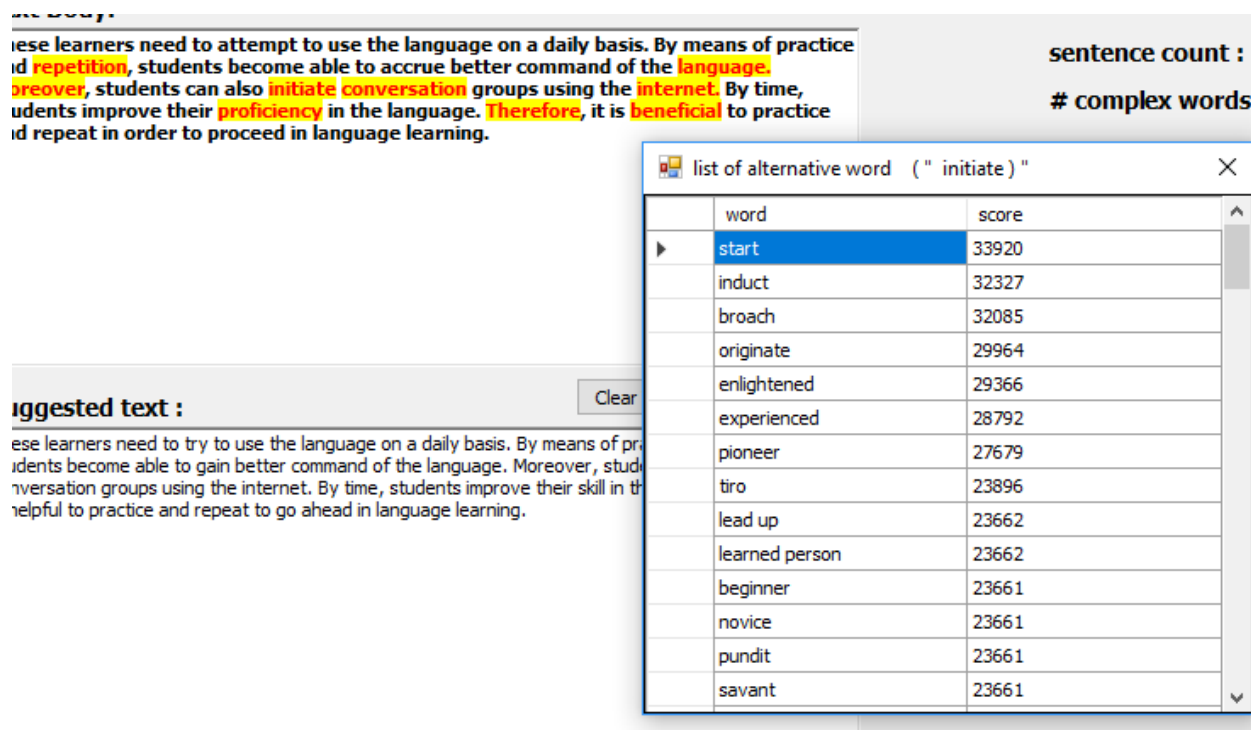
The list of words returned as JSON list with two fields; the first is the alternative word; the second is the score. For this I quote “For queries that have a semantic constraint, results are ordered by an estimate of the strength of the relationship, most to least. Otherwise, queries are ranked by an estimate of the popularity of the word in written text, most to least. At this time, the "score" field has no interpretable meaning, other than as a way to rank the results.”³

The following code is function that used to consume the API and return list in Dataset.

```
Public Sub getlist(word As String)
    Me.Text = "list of alternative word ( "" " & word & " ) """"
    _word = word
    Dim url As String = "https://api.datamuse.com/words?ml=" & word
    Dim strResult As String = String.Empty
    Dim webrequest__1 As HttpWebRequest = DirectCast(WebRequest.Create(url), HttpWebRequest)
    webrequest__1.Method = "GET"
    webrequest__1.ContentType = "application/json"
    Dim webresponse As HttpWebResponse = DirectCast(webrequest__1.GetResponse(), HttpWebResponse)
    Dim enc As Encoding = System.Text.Encoding.GetEncoding("utf-8")
    Dim loResponseStream As New StreamReader(webresponse.GetResponseStream(), enc)
    strResult = loResponseStream.ReadToEnd()
    loResponseStream.Close()
    webresponse.Close()
    'Deserialize the JSon value and assign it to datatable
    Dim dtValue As DataTable = DirectCast(JsonConvert.DeserializeObject(strResult, (GetType(DataTable))), DataTable)
    If dtValue.Rows.Count > 0 Then
        DataGridView1.DataSource = dtValue
    End If
End Sub
```

³ <https://www.datamuse.com/api/> accessed 2-11-2017

When selected any word from text body the following screen will appear, i.e the word we want to change is “initiate” the list as :



These learners need to attempt to use the language on a daily basis. By means of practice and repetition, students become able to accrue better command of the language. Moreover, students can also initiate conversation groups using the internet. By time, students improve their proficiency in the language. Therefore, it is beneficial to practice and repeat in order to proceed in language learning.

sentence count :
complex words

Suggested text :

list of alternative word (" initiate ")

word	score
start	33920
induct	32327
broad	32085
originate	29964
enlightened	29366
experienced	28792
pioneer	27679
tiro	23896
lead up	23662
learned person	23662
beginner	23661
novice	23661
pundit	23661
savant	23661

Figure 4-11 Alternative Words for "initiate"

After loop in text body and get new text in suggested text we can get the final text and past it into the text body text to check the readability level.

4.4 System Testing

Testing is an important phase in developing and our research to help testers and QAs to complete their tasks with short time. Also with less effort by benefit from existence of the comments and documentation of code as their English readability level.

For this phase we use text as testing paragraph which written by Dr.Majdi Abuzahra from English and translation department in BZU. this paragraph was tested by our proposed tool as show in figure 4-12 and retest it by online free calculated text readability using fox index as Figure 4-12 show the result was that the indexing value is closed to each other.

Text Body:	
<p>reiterate in order to learn the language. These learners need to attempt to use the language on a daily basis. By means of practice and repetition, students become able to accrue better command of the language. Moreover, students can also initiate conversation groups using the internet. By time, students improve their proficiency in the language. Therefore, it is beneficial to practice and repeat in order to proceed in language learning.</p>	<p>sentence count : 6</p> <p># complex words : 14</p> <p>word count : 81</p> <p>Calculate Gunning Fog Index : 12.3135802469136</p> <p>Calculate Flesh Kincaid Index : 6.79808518518519</p>

Figure 4-13 testing tool result

Gunning Fog Index

THE GUNNING FOG INDEX IS 12.31

- The number of major punctuation marks, eg. [.,], was
- The number of words was
- The number of 3+ syllable words, highlighted in blue, was

You can edit the numbers above and recalculate

EDITED TEXT

Most educators deem it necessary for language learners to practice and reiterate in order to learn the language[.] These learners need to attempt to use the language on a daily basis[.] By means of practice and repetition, students become able to accrue better command of the language[.] Moreover, students can also initiate conversation groups using the internet[.] By time, students improve their proficiency in the language[.] Therefore, it is beneficial to practice and repeat in order to proceed in language learning[.]

Figure 4-14 testing from another resource <http://gunning-fog-index.com/fog.cgi>

4.5 Conclusion

A tool was created to be used for enhancing text of comments on code, three formulas were used to assess the text and give rank as number indicated the readability level of text. The system of this tool contains two main modules measurement: readability and replacement module. To support the replacement for suitable term we used online API to be updated that gives closely related words as symmetric and usually used.

Chapter Five

Experiment Design

To get the target aim from existence code comments we should be sure that the others who will check our code later be able to understand what we written both comments and code.

In this chapter we go to use survey to measure the difference of text understanding between raw comments (from programmers applications code) and suggested which from the proposed tool. In this sec 5.1 we explain the Design of the Questionnaire, in section 5.2 we define Research Hypothesis, in section 5.3 we view Experiment process.

5.1. Design of the Questionnaire

The survey was divided into two section: first is for general information about the programmers; this section is used to gather data about the participants' experience in programing field. The second is the one by which we will assess the readability level of programmer from different comments. And each comment duplicated as original one that was written by programmers; and another one which is the enhanced from proposed tool (CRS).

Programmer must read each two related comments, and answer the questions which comment was (Has more complex words, More understandable, Took more time to read at all) some of these comments collected from “github” open source projects and reviewed by Dr. Tawfiq Ammar⁴. And one of them was written by Dr. Majdi Abu Zahara⁵.

The survey was designed by google online form. To be able to distribute this questionnaire to developers in different companies and also another copy created to distribute to IT students at BZU. This distribution of survey by sharing link over emails and on online announcement board of such company like Asal Technologies, Proginer, and in academic course broad in Ritaj which is academic portable at BZU.

⁴ Faculty Member, Department of Languages and Translation at BZU

⁵ Faculty Member, Department of Languages and Translation at BZU

To study the user code source readability from understanding text comments, we first searched and thought about factors that may have effects on the code source readability from code comments during the implementation and maintained phases, and we came up with the following main factors to study:

1. Complex words: this factor may affect the readability level of comment text, since the text with complex words could make it not understandable and need developers with high level of English to understand written text without problem.
2. Text understandable: this factor may be considered as main factor of readability aim, because the level of understanding is critical because if the programmer can't understand what was written, this means that the comment existence is not necessary and unhelpful.
3. Time took to read the text of comment: this factor is very important even if the time needed to read comment and understand it takes more time to read source code and understand it; this makes existence of these comments useless.
4. Working with a team: this factor may affect the readability because when some wrote comment for team, he will be care about the readability level of comment to be understandable by his team colleagues.
5. Necessity of comments existence in the code and if code comments help programmer in understanding the code: if programmer care about the comment, he will comment his code by suitable and valuable comments; also every time code changes, he will also update comments.

5.2. Research Hypothesis

Our assumption is to use a tool with text readability assessments to improve the quality of written comment in development phase. Using new composition of three formulas to assess the text. And external API with alternative words resource in addition to local supported list with most complex words in the English language. By this tool, programmers will be able to change complex or not understanding words written in comments that will make these comments readability level suitable for other programmers or partnership to reduce time taken in understanding of code and make this code more readable which affects the code maintainability and reusability.

5.2.1. System hypothesis

Null hypothesis:

There is no significant difference in comment readability level using CRS tool in development phase and comment readability level without of using CRS.

Alternative hypothesis:

There is significant difference in comment readability level using CRS tool in development phase and comment readability level without of using CRS.

Independent variable:

Comments (original comments, new comments from proposed tool).

Dependent variable:

- The understanding level of other programmers to understanding diffident comments.

Conditions:

- Using tool to change comments.
- Comments as from user.

5.3. Survey Participants

Survey was distributed to programmers who work in different companies in addition to 3rd and 4th year students at IT college in Birzeit University campus. We used this Survey to collect the feedback from students (experimenters).

- **Programmers** who work in different companies(ASAL Tech. Comp, Birzeit Computer Center, State of Palestine Ministry of Interior, and another companies),. The sample size was 41 participants. These programmers have different experience levels in addition to different programming language. This variance is good in order to clarify the relevance of the professional and technical level to the importance of the existence of comments, as well as the level of understanding and intended comments, where this is different for each level and languages.
- **IT Students:** Birzeit University students from Faculty of Engineering and Information Technology, undergraduates, specialized in CS with sample size of 35 students. Selection randomly from Computer science students with year level of 3, 4, 5.

We used T-test to analyze the data. For this the participants sample was randomly selected from programmers also from students. Finally, the sample size for programmers was 41 participants and for students were 35 students.

5.4. Conclusion

To assess the result from proposed tool, a survey was created using online google forms to be distributed over our participants mentioned in sec 5.3; the three main questions repeated for each of comments (Necessary of Existence comments in the code, if code comments help programmer in understanding the code, and Text understandable) was the important question used to evaluate the text and compared between the original text and enhanced one. In the next chapter we will discuss the data which collected.

Chapter Six

Data analysis and Discussion

This chapter presents the results of this research survey. In section 6.1 data analysis from participants will be discussed; it also shows the answers for each question in the survey. In section 6.2 we discuss the results and compare with the results from expert programmers and BZU students.

6.1. Data analysis

➤ Programmers.

Within this scope the result from survey was that there are four main languages that developer set as main programming languages: java and its percentage was 17.1%, php with percentage of 19.5%, python with percentage of 12.2%, and .net with percentage of 24.4%. The rest value for 10% language as percentage of 2.4% to 5% for each as the following figure (14) show.

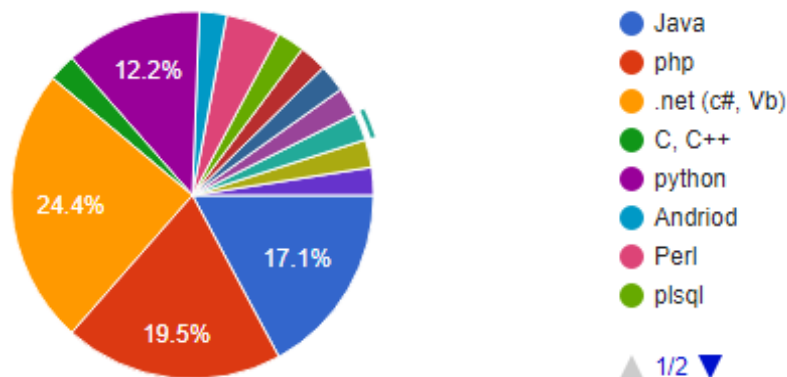


Figure 6-1 Result for primary programming language of participants

The following charts show that most of programmers have very good English level in three parts (reading, writing, and speaking).

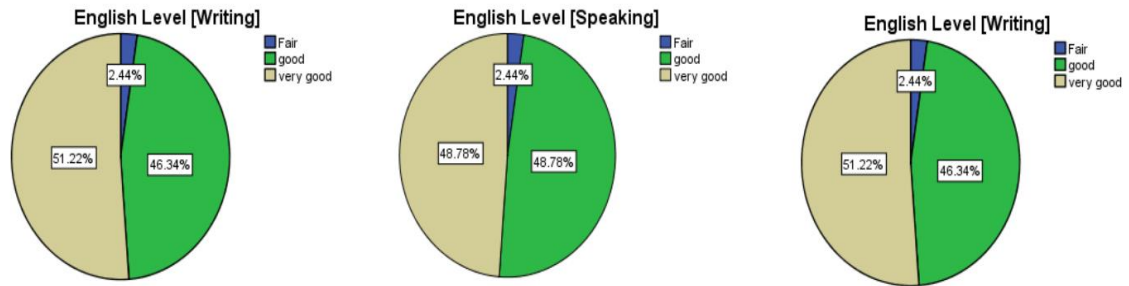


Figure 6-2 Programmer English level

The result about the working in group as a team the result was as 90.24% which means that most of them preferred to work in group rather than work individually.

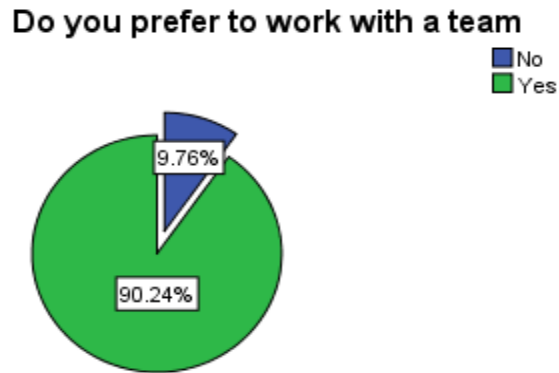


Figure 6-3 working in team result

Also the result about the comments (Did you use comments in coding? Do you think the comments in the code are necessary? And do code comments help you in understanding the code?). After reviewing the overall result was that most programmers encourage others to use comments and they supports the existence of comments in code file.

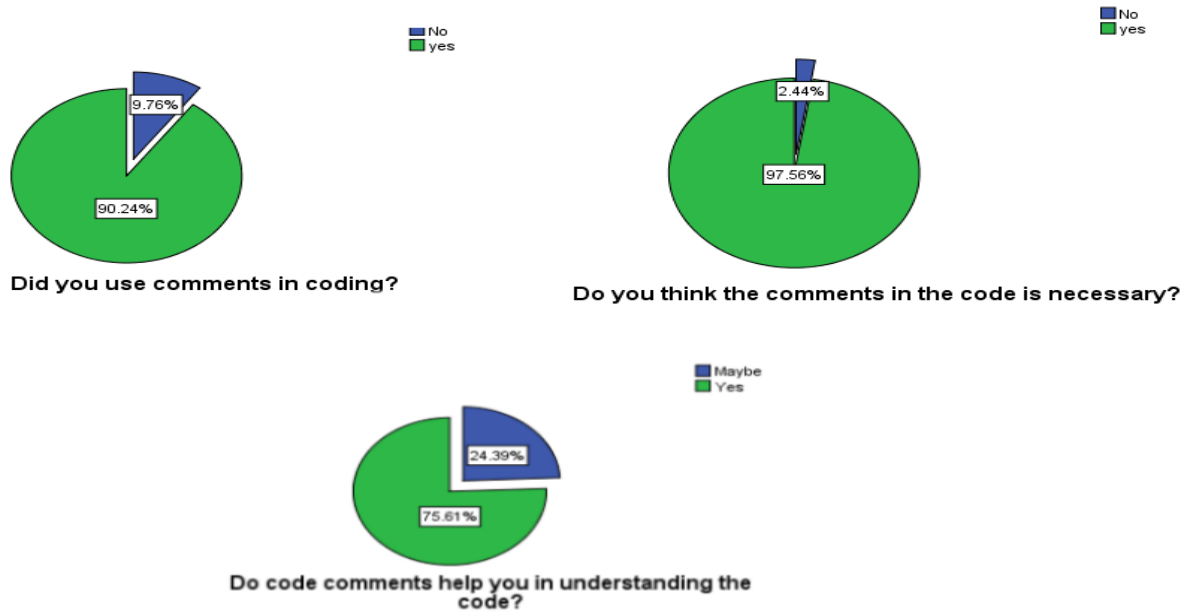


Figure 6-4 Result about source code comments

However, the important part of the survey is about the three questions asked about the original comments which were collected from github projects and the new one, which was the result from our proposed tool. From first question about which of two comments “**Has more complex words**”. The results are shown in figure bellow.

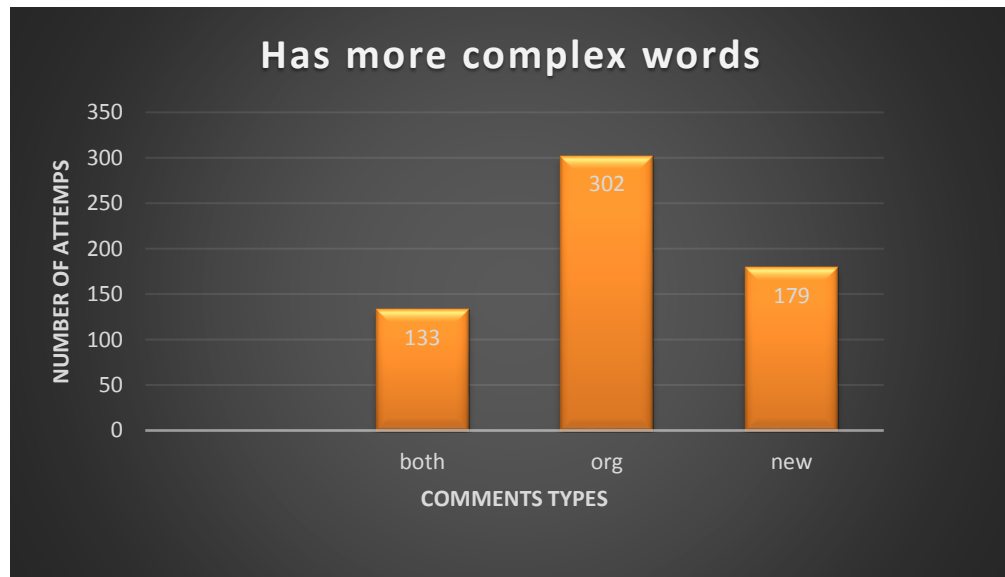


Figure 6-5 chart of result of question: Has more complex words?

From the above figure the replacement function with alternative words makes text words more simple and the result was that the original comments has complex words marked as 49.11%. But the new was

with percentage of 29.11% this means that there is difference of complexity between the two texts as result shown. And this may affect the time of reading the text of comments. And the result was a positive relationship that the factor of number of complex words took more time to read and get the main idea from the text. The figure below shows the result of time that took to read the text of each comments. Programmers go to set that the text of original comments which was mentioned before as having more complex words took more time to read.

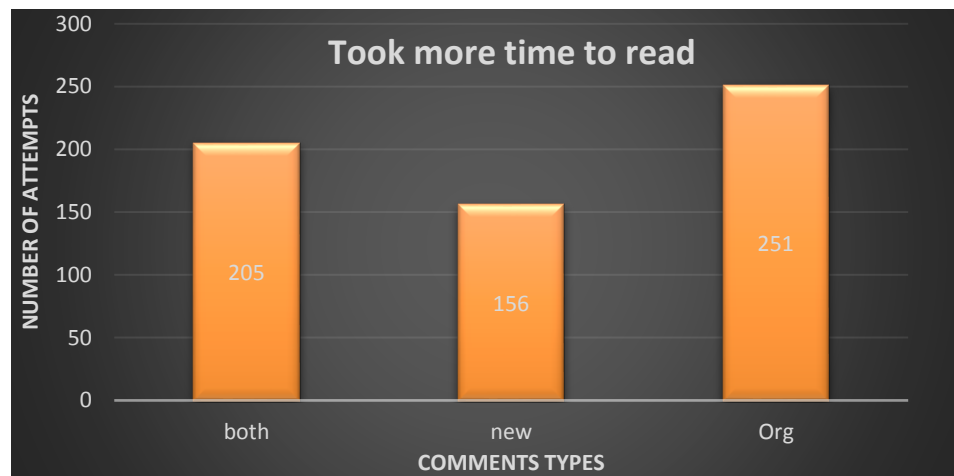


Figure 6-6 result of question: Took more time to read?

Finally about the last question which was: "Which one was more understandable?" The results were almost the two texts were with same level and this is because the advance level of English that the complexity of words did not affect the understandability of reading because these programmers are proficient in English as we discussed earlier and this assumption was result from [30] that says that there a strong correlation between reading comprehension and vocabulary knowledge. The following figure shows this

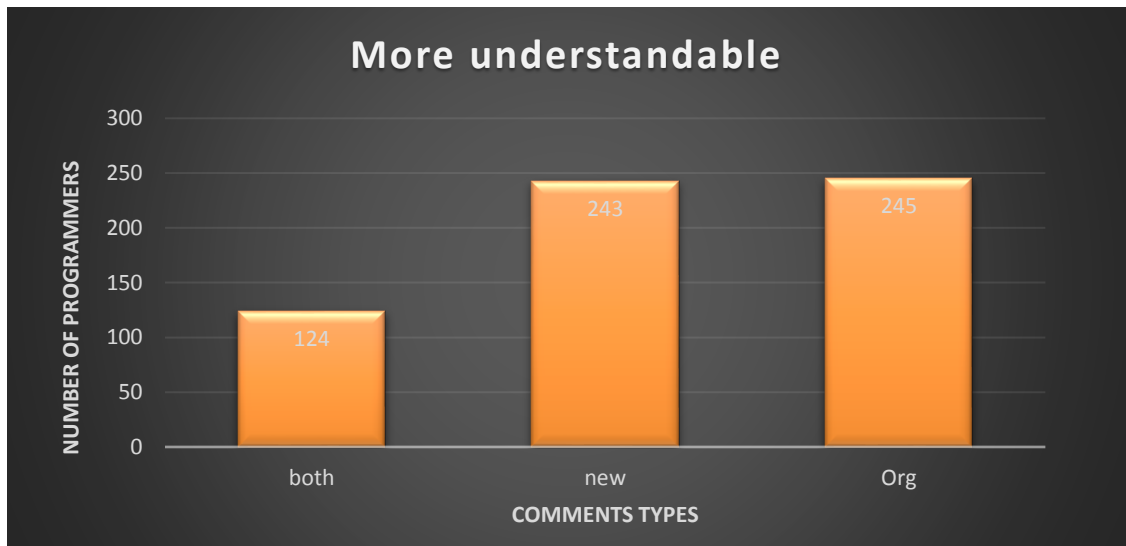


Figure 6-7 result for question: which was more understandable?

- **IT Students:** Birzeit University students from Faculty of Engineering and Information Technology as mentioned in sec 5.3 random selection from Computer science students with year level of 3, 4, 5. the result as year level as figure shows :

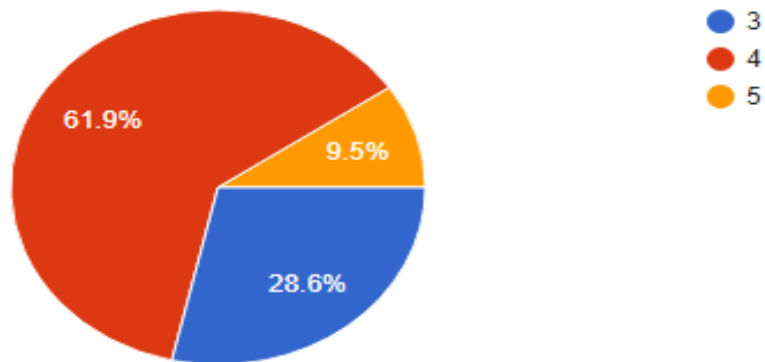


Figure 6-8 student year level

For these students java as programing language takes the most value as 95.2% as primary language

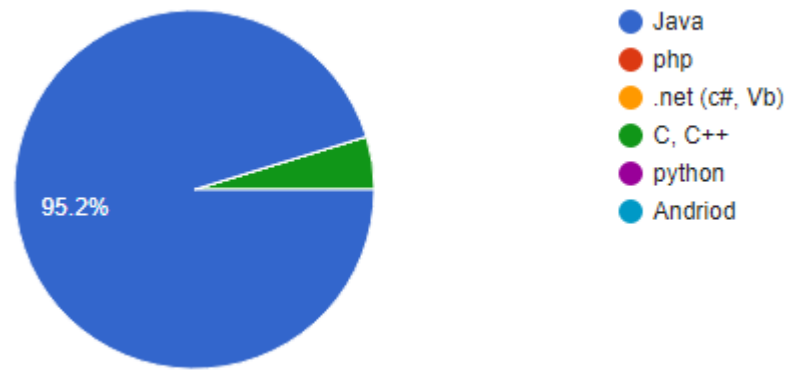


Figure 6-9 student primary programming language

However, going back to the important part of the survey about the three questions asked about the original comments which were collected from github projects and the new ones which were the result from our proposed tool. From first question about which of two comments “**Has more complex words**”. The result is shown in figure bellow.



Figure 6-10 chart of result of question: Has more complex words?

From above figure the replacement function with alternative words make text words more simple and the result was as the original comments has complex words marked as 54%. Whereas the new was with percentage of 29%; this means that there is difference of complexity between the two texts as

result has shown. And this may affect the time of reading the text of comments. And the result was a positive relationship that the factor of number of complex words took more time to read and get the main idea from the text. The figure below shows the result of time that took to read the text of each comments. Students believe that the text of original comments mentioned before has more complex words which took more time to read; the result of this finding was the new comments took less time to read with percentage of 23%, whereas the original was with percentage of 40%.

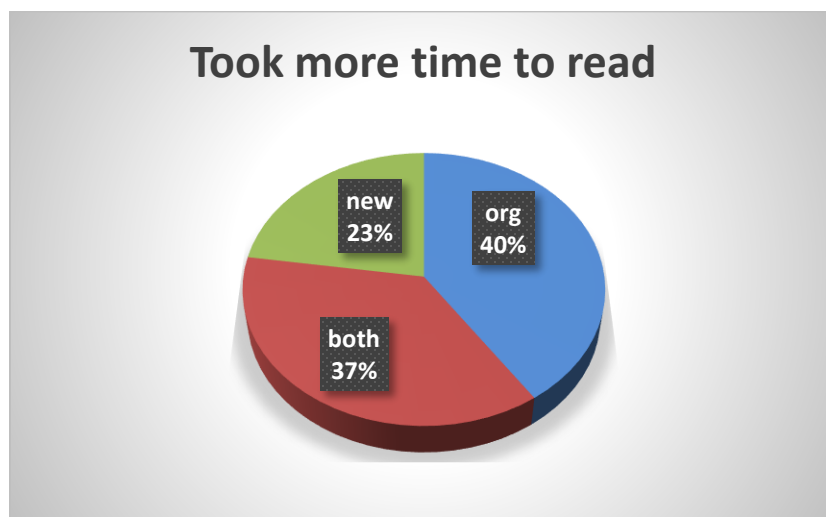


Figure 6-11 result of question: Took more time to read?

Finally about the last question which was: "Which one was more understandable?" .The following figure shows the results were almost the new comments are more understandable with percentage of 56% , but the original comments were 30%;this gives an indicator that the level of English and experience may affect the readability.

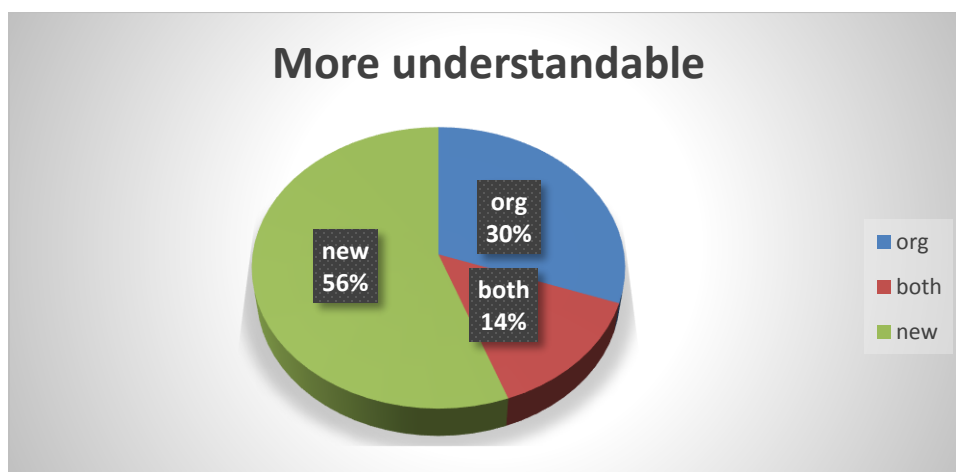


Figure 6-12 result for question: which was more understandable?

6.2. Discussion

For programmers to do data analysis we apply T-test on data to approved research hypothesis in measure readability. The result as shown in the following table with 95% Confidence Interval of the Difference that there is no significant difference for understanding factor of new text compared with original text because the value was (0.969) and this value is more than .05; this result may not be affected by text complexity because of the good English level of programmers in our random sample. But the result of variable of text complexity (text with more complex words) the result was (0.00) less than (0.05); this means that there is significant difference between the original text and new text. And this evidence supported research hypothesis that the CRS with replacing module reduce text complexity level and t value was -4.458; this means the new text contains less complex words than the original one. Finally, the last measured variable was the time taken to read text; the result value was (0.01) and this is less than 0.05 which means there is significant difference between compared comments; also the t value is -3.788, which means that the new text took less time to read than the original one. Thus this supported our assumption about text readability to reduce the time to understand the comments on source code. Finally, we can say there is significant difference in comment readability level using CRS tool in development phase and comment readability level without using CRS and this significance was that the new one contains less number of complex words and took less time to read.

One-Sample Statistics				
	N	Mean	Std. Deviation	Std. Error Mean
more_understandable	41	.0016	.26977	.04213
more_complex	41	-.2016	.28963	.04523
more_time_to_read	41	-.1512	.25561	.03992

One-Sample Test						
	Test Value = 0					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
more_understandable	.039	40	.969	.00163	-.0835	.0868
more_complex	-4.458	40	.000	-.20163	-.2930	-.1102
more_time_to_read	-3.788	40	.001	-.15122	-.2319	-.0705

Figure 6-13 programmers data new comment readability t-test values

Also we apply test correlation type between three questions to check if our hypothesis is true or the alternative is true. As the table below shows, there is relation between the three variables we search for (number of complex words, text understanding, and the time taken to understand the whole text and get

the idea from it). The results support the research assumption that the text with more complex words takes more time to read vice versa. Also the following table shows each variable with others two variables. If we study the relation between understanding text and containing number of complex words the relation was revised relation (more understandable the less complex words and vice versa). On the other hand, more time to read this means text contains more complex words and more time means less understanding; this is what the study is trying to prove.

Correlations				
		more_complex	more_understandable	more_time_to_read
more_complex	Pearson Correlation	1	-.674**	.532**
	Sig. (1-tailed)		.000	.000
	N	41	41	41
more_understandable	Pearson Correlation	-.674**	1	-.497**
	Sig. (1-tailed)	.000		.000
	N	41	41	41
more_time_to_read	Pearson Correlation	.532**	-.497**	1
	Sig. (1-tailed)	.000	.000	
	N	41	41	41
**. Correlation is significant at the 0.01 level (1-tailed).				

Table 6-1 result of correlation between three main questions

For students data analysis we apply T-test on data to prove research hypothesis in measure readability. The result as shown in the following table with 95% Confidence Interval of the Difference that there is significant difference for understanding variable of new text compared with original text because the value was (0.000) and this value less than (0.05), also the t value was (5.066) this means that new comment is more understandable than original one. Another variable was text complexity (text with more complex words) the result was (0.00) less than (0.05) this means that there is significant difference between the original text and new text. And this an evidence supported research hypothesis that the CRS with replacing module reduce text complexity level and t value was (-4.695); this means the new text contains less complex words than original one. Finally, the last measured variable was the time that it took to read text; the result value was (0.01) and this is less than 0.05 which means there is significant difference between compared comments also the t value is -3.837 ;this means that the new text took less time to read than original one. Thus this supported our assumption about text readability

to reduce the time to understand the comments on source code. Finally we can say there is significant difference in comment readability level using CRS tool in development phase and comment readability level without using CRS and this significant was that new one is more readable than the original text of comments.

One-Sample Statistics				
	N	Mean	Std. Deviation	Std. Error Mean
new_has_more_complex words	35	-.2438	.30719	.05192
new_has_more_understandable	35	.2343	.27362	.04625
new_has_more_time_to_read	35	-.1695	.26139	.04418

One-Sample Test						
	Test Value = 0					
	t	df	Sig. (2-tailed)	Mean Difference	95% Confidence Interval of the Difference	
					Lower	Upper
new_has_more_complex words	-4.695	34	.000	-.24381	-.3493	-.1383
new_has_more_understandable	5.066	34	.000	.23429	.1403	.3283
new_has_more_time_to_read	-3.837	34	.001	-.16952	-.2593	-.0797

Figure 6-14 student data new comment readability t-test values.

Also we apply test correlation type between three questions to check if our hypothesis is true or the alternative is true. As the table below shows there is significant relation between the three variables we search for (number of complex words, text understanding, and the time that take to understood the whole of text and get the idea from it) because Correlation is significant at the 0.01 level. These results supported the research assumption that the text with more complex words takes more time to read and also the about understanding it was inverse relationship. Also the following figure shows each variable with other two variables. If we study the relation between understanding text and containing number of complex words the relation was revised relation (more understandable the less complex words and vice versa). On the other hand, more time to read this means text contains more complex words and more time means less understanding and this is what this research attempts to prove.

Descriptive Statistics			
	Mean	Std. Deviation	N
new_has_more_complex words	-.2438	.30719	35
new_has_more_understandable	.2343	.27362	35
new_has_more_time_to_read	-.1695	.26139	35

Correlations				
		new_has_more_complex words	new_has_more_understandable	new_has_more_time_to_read
new_has_more_complex words	Pearson Correlation	1	-.699**	.639**
	Sig. (2-tailed)		.000	.000
	N	35	35	35
new_has_more_understandable	Pearson Correlation	-.699**	1	-.644**
	Sig. (2-tailed)	.000		.000
	N	35	35	35
new_has_more_time_to_read	Pearson Correlation	.639**	-.644**	1
	Sig. (2-tailed)	.000	.000	
	N	35	35	35

** . Correlation is significant at the 0.01 level (2-tailed).

Figure 6-15 result of correlation between three main questions

6.3. Conclusion

After the data analysis of two experimenters (programmers and students) we found that students were benefited more from the modification and enhancement. And that they have expressed satisfaction with the new text improved, and that if this pointed to something it indicates that the assumption on which the research is based is correct, especially that there is a difference in the levels of English language if compared with programmers. Hence, the CRS which is used has actually improved the level of readability and people who have a language problems or those not familiar with complex or unusual words can use it, to improve the existing comments or documentations readability level especially that is this system is supported by an updated API.

Chapter Seven

Conclusion and Future Work

In this chapter section 7.1 we discuss the conclusion of our research, and in the section 7.2 listed the future work that maybe help others to continue and upgrade this work.

7.1. Conclusion

The Quality code depends on many factors that influence its readability; one of them is comment (the part which describes the code and gives more information about the section which is written about. Therefore, this comment is written to be used by other developers and developers who write the code, for this issue these comments should be readable and easy to be understood by others. Therefore, we measure the comments readability and make some changes to its terms. By this change we aim to make its readability level suitable for others. These changes will contribute to achieving the target readability level, also they will make code understanding easier. Code understanding means the code readability is easy so the maintainability and reusability become easier also.

We implement a tool to assess source code comments readability, this tool used three famous formulas (Fog index, Flesch reading ease score, and Flesch-Kincaid grade level) to measure text readability, also used two resources for words and terms alternatives. We used an updated API to be sure about the alternative words choices is updated and closer to required term. We collected comments from github projects then reviewed them by an instructor at Department of Languages and Translation at BZU, Dr. TAWFIQ Ammar. All of these comments are complex and unusual used words replaced by terms with less complexity or familiar to most of people. However, this change affected the readability level for each of the comments. This was evaluated by creating a survey completed by 41 senior programmers in addition to 35 students. The survey consisted of two sections: first one was about general information about the experimenters. And the second one consists of 15 questions, each question contains two comments, one from github projects and another one was from the enhanced comment by our proposed tool and experimenter should answer which of two comments (are more has complex words, are more understandable, and which one took more time to read). Survey participants as programmers agreed that enhanced comments contain less number of complex words as percentages of (49% for original

comments, 29% new comments have more complex words, 22% there is no difference, also these comments were more understandable than original ones as percentages of (40% for original comments, 40% new comments have more complex words, 20% there is no difference). and finally enhanced comments took less time to read as percentages of (41% for original comments, 25% new comments have more complex words, 34% there is no difference). With previous information and from t-test analysis the result shows that the new comments were better in two variables (the new comments contain more complex words, and if new comment took more time to read) but the result of understating was that there is no difference between two compared comments. We guess that this was the high level of English of expert programmers that the complex word did not affect the understanding of text.

For Survey participants as students were agreed that enhanced comments contain less number of complex words as percentages of (54% for original comments, 29% new comments have more complex words, 17% there is no difference), also these comments were more understandable than original one as percentages of (30% for original comments, 56% new comments has more complex words, 14% there is no deference). And finally enhanced comments took less time to read as percentages of (40% for original comments, 23% new comments have more complex words, 37% there is no difference). With previous information and from t-test analysis the results show that the new comments were better in three concerned variables (is new more understating, is new contains more complex words, and if new comment took more time to read) which were different from programmers where the understanding of text was not affected and we guess as mentioned before the high level of English language.

From previous discussions we can say that the tool which we built may help people for whom English is not the mother tongue rather than those who have very good level in English or those who are native speakers of English.

7.2. Suggestions for future work

In the future there are many upgrades that can be done to this system:

- 1- Add this tool as add on to IDE; this will make reading comments more easier and help programmers to change the complex word while writing own comments.
- 2- Connected tool to corpus with other languages. Because we focus on English and make alternative words from list retrieved from this corpus.

- 3- Add spellchecking that may help programmers to reduce the error that come up from typo mistake.
- 4- Add more terms to local languages. That should be used in future as local corpus.
- 5- Add NLP smart metrics to change term with suitable alternative terms.
- 6- Add new feature to consume the comments automatically from projects and combined as enhanced documentation.

References

- [1] A. Saha, "Origins of poor code readability ," in *PPIG 2011 - 23rd Annual Workshop*, University of York, 2011.
- [2] H. Aman, S. Amasaki, T. Yokogawa and M. Kawahara, "Empirical analysis of words in comments written for Java methods," in *Proceedings - 43rd Euromicro Conference on Software Engineering and Advanced Applications, SEAA 2017*, 2017.
- [3] R. P. Buse and W. R. Weimer, "Learning a metric for code readability," *IEEE Transactions on Software Engineering*, 2010.
- [4] M. Agrawal and K. Chari, "Software effort, quality, and cycle time: A study of CMM level 5 projects," *IEEE Transactions on Software Engineering*, 2007.
- [5] R. D. Finlayson, N. M. Mitsumori and F. X. Reddington, "System to Monitor and Maintain Balance of Factory Quality Attributes Within a Software Factory Operating Environment," *U.S. Patent Application 11/844*, p. 31, 26 February 2009.
- [6] "umbrella_defs," [Online]. Available: http://www.hq.nasa.gov/office/codeq/software/umbrella_defs.htm . [Accessed 2015 11 2015].
- [7] D. Schreck, V. Dallmeier and T. Zimmermann, "How Documentation Evolves over Time," *Ninth international workshop on Principles of software evolution in conjunction with the 6th ESEC/FSE joint meeting - IWPSE '07*, 2007.
- [8] D. Steidl, B. Hummel and E. Juergens, "Quality analysis of source code comments," in *2013 21st International Conference on Program Comprehension (ICPC)*, 2013.
- [9] A. Vikström, "A study of automatic translation of MATLAB code to C code using software from the MathWorks," 2009.
- [10] A. Batool, M. H. u. Rehman, A. Khan and A. Azeem, "Impact and comparison of programming constructs on java and c# source code readability," *International Journal of Software Engineering and its Applications*, 2015.
- [11] "Flesch–Kincaid readability tests," Wikipedia, [Online]. Available: https://en.wikipedia.org/wiki/Flesch–Kincaid_readability_tests. [Accessed 18 10 2016].
- [12] B. Spuida and S. W. Wrangler, "The fine Art of Commenting," 2002.
- [13] D. Tobias, "Continuous Code Reviews: A Social Coding tool for Code Reviews inside the IDE," *In Companion to the first International Conference on the Art, Science and Engineering of Programming*, p. 14, 2017.
- [14] S. U. Ahmed and P. Rajendra , "Evaluating Efficiency and Effectiveness of Code Reading Technique with an Emphasis on Enhancing Software Quality," in *National Conference On Advances In Technology & Applied Sciences*, 2014.
- [15] J. Cohen, "Reviewing Code with Perforce".
- [16] I. E. Akpannah, "Optimization of Software Quality using Management and Technical Review Techniques," vol. 17, 2014.

- [17] N. Khamis, R. Witte and J. Rilling, "Automatic quality assessment of source code comments: The JavadocMiner," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010.
- [18] Y. Tashtoush, Z. Odat, I. Alsmadi and M. Yatim, "Impact of programming features on code readability," *International Journal of Software Engineering and its Applications*, 2013.
- [19] "XML Documentation Comments (C# Programming Guide)," [Online]. Available: <https://msdn.microsoft.com/en-us/library/b2s063f7.aspx..> [Accessed 23 12 2017].
- [20] R. Namani and K. J, "A New Metric for Code Readability," *IOSR Journal of Computer Engineering* , vol. 6, no. 6, pp. 44-48, 2012.
- [21] V. Podugu, "developing a code readability model to improve software quality," *International Journal of Computer Science & Informatic*, vol. 1, no. 2, 2011.
- [22] G. Amrulla, M. Mourya, A. Ahad Afroz and S. Kashif Ali, "A Survey of Improving Computer Program Readability to Aid Modification," *International Journal of Advanced Trends in Computer Science and Engineering*, 2014.
- [23] D. Boswell and T. Foucher, *The art of readable code.*, O'Reilly Media, Inc, 2012.
- [24] R. Stoll, *Java Code Conventions*, 2014.
- [25] P. Goodliffe, *Code Craft*, 2007.
- [26] C. Hendrickson, " CHAPTER 2:Quantitative measures of text complexity," in *TEXT COMPLEXITY*, 2016, pp. 25-38.
- [27] A. Corazza, V. Maggio and G. Scanniello., "Coherence of comments and method implementations: a dataset and an empirical investigation.,", *Software Quality Journal*, pp. 1-27, 2016.
- [28] E. Davoodi and a. L. Kosseim., "On the Contribution of Discourse Structure on Text Complexity Assessment.," *Computation and Language*, pp. 166-174, 19 Aug 2017.
- [29] H. Aman, T. Sasaki, S. Amasaki and M. Kawahara, "Empirical analysis of comments and fault-proneness in methods," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement - ESEM '14*, 2014.
- [30] K. H. Mühlenbock, S. J. Kokkinakis, C. Liberg, Å. Geijerst, J. W. Folkeryd, A. Jönsson, E. Kanebrant and J. Falkenjack, "A multivariate model for classifying texts' readability," *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA)*, 2015.
- [31] M. Zamanian and P. Heydari, "Readability of Texts: State of the Art," *Theory and Practice in Language Studies*, 2012.
- [32] M. Smith and R. Taffler, "Readability and Understandability: Different Measures of the Textual Complexity of Accounting Narrative," *Accounting, Auditing & Accountability Journal*, 1992.
- [33] K. M. Sheehan, "Validating Automated Measures of Text Complexity," *Educational Measurement: Issues and Practice*, 2017.
- [34] L. Feng, M. Jansche, M. Huenerfauth and N. Elhadad, "A comparison of features for automatic readability assessment," *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, 2010.
- [35] E. Collar and R. Valerdi, "Role of Software Readability on Software Development Cost," *In Proceedings of the 21st Forum on COCOMO and Software Cost Modeling*, 2006.

[36] L. Feng, Automatic readability assessment., City University of New York, 2010.

Appendix

Table of Figures:

Figure 1-1 quote about importance of comments	1
Figure 2-1 “The iron triangle” managing the balance between business transformation and risk [5]....	5
Figure 2-2: Opportunities to optimize quality and time to market exist throughout the application development lifecycle [5].	5
Figure 2-3 Code Should Be Easy to Understand [23].....	10
Figure 2-4 is comments existence necessary after a while [23].....	11
Figure 2-5 when comment is needs [23].....	12
Figure 4-1 system general steps	23
Figure 4-2 overall system process.....	23
Figure 4-3 Module Functions and process.....	25
Figure 4-4 Figure of main function of getting indexval from Fog Index equation is in “sec 3.1”	26
Figure 4-5 main Screen of system	28
Figure 4-6 system calculations section view	28
Figure 4-7 system complex words list view	29
Figure 4-8 sample complex words with simple alternative	29
Figure 4-9 code for Replacing terms from local database	30
Figure 4-10 example of using local database for replacement the terms.....	30
Figure 4-11 Alternative Words for "initiate"	32
For this phase we use text as testing paragraph which written by Dr.Majdi Abuzahra from English and translation department in BZU. this paragraph was tested by our proposed tool as show in figure 4-12 and retest it by online free calculated text readability using fox index as Figure 4-12 show the result was that the indexing value is closed to each other.	32
Figure 4-13 testing tool result.....	33
Figure 4-14 testing from another resource http://gunning-fog-index.com/fog.cgi	33
Figure 6-1 Result for primary programming language of participants	38
Figure 6-2 Programmer English level.....	39
Figure 6-3 working in team result.....	39
Figure 6-4 Result about source code comments	40
Figure 6-5 chart of result of question: Has more complex words?.....	40
Figure 6-6 result of question: Took more time to read?	41
Figure 6-7 result for question: which was more understandable?	42
Figure 6-8 student year level.....	42
Figure 6-9 student primary programming language	43
Figure 6-10 chart of result of question: Has more complex words?.....	43
Figure 6-11 result of question: Took more time to read?	44

Figure 6-12 result for question: which was more understandable?	44
Figure 6-13 programmers data new comment readability t-test values	45
Figure 6-14 student data new comment readability t-test values.....	47
Figure 6-15 result of correlation between three main questions.....	48

Table of Tables:

Table 3-1 flesch reading ease score to assess the ease of readability in a document [27]	17
Table 3-2 Formula of Flesch-Kincaid [4].	18
Table 3-3 gunning's fog-index level [27]	19
Table 6-1 result of correlation between three main questions	46

Questionnaires data:

Programmers Questionnaire first part:

English Level [Reading]	English Level [Writing]	English Level [Speaking]	Your Company name	Your Job title	Do you prefer to work with a team	Your primary programming language	Did you use comments in coding?	Do you think the comments in the code are necessary?	Do code comments help you in understanding the code?
good	good	good	Birzeit University	Programmer	Yes	php	yes	yes	Yes
very good	very good	good			Yes	plsql	yes	yes	Yes
Fair	Fair	Fair	ASAL	Software Engineer	Yes	python	yes	No	Maybe
very good	good	very good	Allam Al Qudu	Technical Team Lesder	Yes	php	yes	yes	Yes
very good	very good	very good	ASAL	QA Engineer	Yes	C, C++	yes	yes	Yes
very good	good	good	Palestinians land authority	computer engineer	Yes	.net (c#, Vb)	yes	yes	Yes
very good	very good	very good	Birzeit University	Network & Devices Specialist	Yes	Java	yes	yes	Yes
very good	very good	very good	Cairo amman bank	Non banking system officer	Yes	php	yes	yes	Yes
very good	good	very good	Ministry of	IT Head	Yes	.net (c#, Vb)	yes	yes	Yes

			Women's Affairs						
very good	very good	very good	Asal Technologies	Software Engineer	Yes	python	yes	yes	Yes
very good	very good	very good	Asal Technologies	Software Engineer	Yes	python	yes	yes	Yes
very good	very good	very good		networking manager	Yes	php	No	yes	Maybe
good	good	good	Asal	SE	Yes	Perl	yes	yes	Yes
very good	very good	very good	Asal Tech	SF Developer	Yes	Apex, Visualforce	yes	yes	Yes
very good	very good	very good	Birzeit university	Developer	No	Java	yes	yes	Yes
very good	good	good	ProGineer	Engineering	Yes	Perl	yes	yes	Yes
good	good	good	ProGineer Technologies	Software Developer	Yes	.net (c#, Vb)	yes	yes	Yes
very good	very good	very good			Yes	.net (c#, Vb)	yes	yes	Yes
very good	very good	very good	Exalt	Software Engineer	Yes	JavaScript	yes	yes	Yes
good	good	good	Dimensions	Full-stack developer	Yes	Java	yes	yes	Maybe
good	good	good	BZU	software eng	No	php	yes	yes	Yes
very good	very good	good	Dimensions Info Tech	Senior System Developer	Yes	.net (c#, Vb)	yes	yes	Maybe
good	good	good	dimensions-infotech	Developer	Yes	.net (c#, Vb)	No	yes	Maybe

very good	very good	very good	TetraTeck DPK	Development Manager	Yes	Java	yes	yes	Yes
good	good	good	Birzeit University	Systems Specialist	Yes	.net (c#, Vb)	yes	yes	Yes
very good	very good	good	Birzeit University	IT Infrastructure Head	Yes	Groovy, ruby	yes	yes	Yes
good	good	good	JPMorgan Chase	software engineer	Yes	.net (c#, Vb)	No	yes	Yes
very good	very good	very good		student	Yes	python	yes	yes	Yes
very good	very good	very good	Exalt	Technical Lead and Architect	Yes	Java	yes	yes	Yes
good	good	very good	MOI	Senior Developer	Yes	.net (c#, Vb)	yes	yes	Yes
very good	very good	very good	BZU	developer	No	tcl	No	yes	Maybe
very good	good	good	Sanabel	Web Engineer	Yes	Java	yes	yes	Yes
very good	very good	very good	Najah	Student	Yes	Andriod	yes	yes	Maybe
very good	very good	very good	BirZeit University	Assistant Professor	Yes	Java	yes	yes	Maybe
very good	very good	very good	BCI	Technical Service Manager	Yes	php	yes	yes	Yes
very good	good	good	Education sector	Head of system a analysis section	No	python	yes	yes	Maybe
good	good	good	An-Najah National University	Teaching assistant	Yes	php	yes	yes	Yes

very good	very good	good	Experts	Oracle ERP Developer	Yes	Oracle	yes	yes	Yes
good	good	good	Harri	QA	Yes	.net (c#, Vb)	yes	yes	Maybe
good	good	good	JT	Full stack Eng	Yes	php	yes	yes	Yes
good	good	good			Yes	DB Admin	yes	yes	Yes

Programmers Questionnaire part 2.1:

Q7 [Took more time to read at all]	Bot h																		
Q7 [More understandable]	Bot h																		
Q7 [Has more complex words]	Bot h																		
Q6 [Took more time to read at all]	C2																		
Q6 [More understandable]	C2																		
Q6 [Has more complex words]	C2																		
Q5 [Took more time to read at all]	C1																		
Q5 [More understandable]	C1																		
Q5 [Has more complex words]	C2																		
Q4 [Took more time to read at all]	C1																		
Q4 [More understandable]	C2																		
Q4 [Has more complex words]	C1																		
Q3 [Took more time to read at all]	C2																		
Q3 [More understandable]	C2																		
Q3 [Has more complex words]	C2																		
Q2 [Took more time to read at all]	Bot h																		
Q2 [More understandable]	Bot h																		
Q2 [Has more complex words]	C2																		
Q1 [Took more time to read at all]	Bot h																		
Q1 [More understandable]	C2																		
Q1 [Has more complex words]	C2																		

C2	C1	Bot h	C2	C1	C2	C1	C2	C1	C1	C2	C1	C1	C2	C1	C2	C1	Bot h	C1	C2	C1
C2	C1	C2	C1	C2	C1	C1	C2	C1	C2	C1	C2	C1	C2	C1	C1	C2	C1	C2	C1	C2
C1	C2	Bot h	C2	C1	Bot h	Bot h	Bot h	Bot h	C1	C2	C1	C1	C1	C2	C2	C1	C2	C1	C2	C2
C2	C1	C2	C1	C2	C1	Bot h	C2	C1	C2	C1	C2	Bot h	C1	Bot h	C2	C1	C2	C2	C1	C1
Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h
C2	C1	C2	C1	C2	C1	C1	C2	C1	Bot h	C2	C1	Bot h	C2	C1	Bot h	C2	Bot h	Bot h	C1	C2
C1	C2	C1	C2	C1	C2	C1	C2	C1	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2
C1	C2	C2	C2	C1	C2	C2	C1	C2	C2	C1	C2	C1	C2	C2	C2	C1	Bot h	C1	C2	Bot h
C2	C1	Bot h	C2	C1	C1	C1	C1	C2	C2	C2	C1	C2	C1	C2	Bot h	Bot h	Bot h	C1	C2	C2
C1	C2	C1	C2	Bot h	C2		C2	C1	Bot h	Bot h	Bot h	C1	C2	C1	C2	C2	C2	Bot h	C1	Bot h
C2	C1	Bot h	C2	C2	Bot h	C2	C1	C2	C2	C2	C2	C2	C2	Bot h	C2	C1	Bot h	C2	C1	C2
C2	Bot h	C2	C1	Bot h	C1	C1	Bot h	C1	C2	Bot h	C2	C2	Bot h	C2	C2	Bot h	C2	C2	Bot h	C2
C2	Bot h	Bot h	C1	Bot h	Bot h	C1	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	C2	Bot h	Bot h	Bot h
C2	C1	C2	C2	C1	Bot h	C1	C2	C1	C1	C2	C1	C2	C1	C1	C1	C2	C2	C1	C2	Bot h
C2	C1	C2	C2	C2	C1	C1	C2	Bot h	C2	C2	Bot h	C2	C2	Bot h	C1	C1	Bot h	C1	C1	Bot h
Bot h	C2	C1	Bot h	C2	C1	Bot h	C2	C1	Bot h	C2	C1	Bot h	C1	C2	Bot h	C1	C2	Bot h	C2	C1
Bot h	Bot h	Bot h	C1	C2	C1	C1	Bot h	C1	C2	Bot h	C2	C2	Bot h	C1	C2	Bot h	C2	C2	Bot h	C2
C2	C1	C2	C1	C2	C2	C1	C2	C1	C2	C1	C2	C1	C2	C2	C2	C1	C2	C1	C2	C1
C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C2	C1	C2	Bot h
C2	C1	Bot h	C1	C2	Bot h	C2	C2	C2	C2	C1	Bot h	C2	Bot h	Bot h	C1	C2	Bot h	C1	C2	Bot h
C2	C2	Bot h	Bot h	Bot h	Bot h	C1	C2	Bot h	C2	C1	Bot h	C1	C1	Bot h	C2	C2	Bot h	C1	C2	Bot h
Bot h	C2	Bot h	Bot h	C2	Bot h	Bot h	C2	Bot h	Bot h	C2	C1	C1	C2	C1	Bot h	C2	Bot h	C1	C2	C1
C2	C1	C1	C2	C1	C2	C1	C2	C1	C2	C2	C2	C1	C2	C1	C1	C2	C2	C1	C1	C2

C2	C1	C2	C2	C2	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C2	C1	C2
C2	C1	C2	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1
C2	C1	C1	Bot h	C2	C2	C1	C2	C1	C2	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	C2	C1	Bot h	Bot h	Bot h
C1	C2	C2	C1	C2	C1	C1	C1	C1	C2	C2	C2	C1	C2	C2	C2	C2	C1	Bot h	C2	C2
C2	C1	C2	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1
C2	C1	Bot h	C2	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	C1	C2	C1	C1	C2	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h
C1	C2	Bot h	C2	C1	Bot h	Bot h	C2	C1	C1	C2	Bot h	C1	C2	C1	Bot h	C2	Bot h	Bot h	Bot h	Bot h
C2	C1	Bot h	C2	C1	C2	C1	C2	Bot h	C1	C2	C2	C1	C1	C2	C2	C1	C2	C1	C2	C2
Bot h	C1	C2	C1	C2	Bot h	C1	C2	C1	C1	C2	C1	C1	C2	C1	C2	Bot h	C1	C1	C2	C1
C2	C1	C2	C1	C2	C1	Bot h	C2	C1	Bot h	C2	C1	C1	C2	C1	Bot h	C2	C1	C1	C2	C1
C2	C1	Bot h	C1	C2	Bot h	C1	C1	Bot h	C2	C2	Bot h	C2	C2	Bot h	C1	C2	Bot h	C1	C2	Bot h
C1	C2	C1	C1	C2	C1	C1	C2	C1	Bot h	C1	Bot h	C2	C2	C1	C2	C1	C2	C1	C2	Bot h
C2	C1	C2	C2	C1	C2	C1	C2	Bot h	C1	Bot h	Bot h	C1	Bot h	Bot h	Bot h	C1	C2	C1	C2	C1

Programmers Questionnaire part 2.2:

Q15 [Took more time to read at all]
Q15 [More understandable]
Q15 [Has more complex words]
Q14 [Took more time to read at all]
Q14 [More understandable]
Q14 [Has more complex words]
Q13 [Took more time to read at all]
Q13 [More understandable]
Q13 [Has more complex words]
Q12 [Took more time to read at all]
Q12 [More understandable]
Q12 [Has more complex words]
Q11 [Took more time to read at all]
Q11 [More understandable]
Q11 [Has more complex words]
Q10 [Took more time to read at all]
Q10 [More understandable]
Q10 [Has more complex words]
Q9 [Took more time to read at all]
Q9 [More understandable]
Q9 [Has more complex words]
Q8 [Took more time to read at all]
Q8 [More understandable]
Q8 [Has more complex words]

C1	C2	C1	Bo th	C1	Bo th	C2	C1	C2	Bo th	Bo th	Bo th	C1	C2	Bo th	Bo th	C2	C1	C1	C2	C1	Bo th	C2	Bo th
C1	C2	C1	C2	C1	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C1	C2	C2	C1	C2
C2	C2	C2	C1	C2	Bo th	C1	C1	C1	C2	C2	C1	C1	C1	C1	C1	C1	C1	C1	C1	C2	C1	C2	C1
Bo th	Bo th	C1	Bo th	Bo th	C1	Bo th	Bo th	C2	Bo th	Bo th	C2	Bo th	Bo th	C1	Bo th	Bo th	C1	Bo th	Bo th	C1	Bo th	Bo th	C2
Bo th	Bo th	Bo th	C2	Bo th	Bo th	C1	C2	C1	C1	C2	Bo th	C2	Bo th	C2	C1	C2		C2	C1	C2	C1	C2	C1
C2	C1	C2	C1	C2	C1	C2	Bo th	C2	C2	C1	C2	C1	C2	C1	C1	C2	C2	C1	C2	C1	C1	C2	Bo th
C2	C1	C2	C2	C1	C2	C2	C1	C2	C2	C1	C2	C1	C2	C1	C2	C1	C2	C2	C1	C2	C2	C1	C2
Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	C1	C2	Bo th	C2	C1	C2	C1	C2	C1	C1	C2	Bo th	C2	C1	Bo th	C1	C2	C1
C1	C1	Bo th	C2	C2	C1	C2	C1	C2	C2	C1	C2	C2	C1	C2	C1	C2	Bo th	C1	C1	C2	C1	Bo th	Bo th
Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th
Bo th	C2	C1	Bo th	C1	C1	C2	C1	Bo th	C1	C2	Bo th	C2	C2	C2	Bo th	C1	Bo th	C2	C1	C2	C1	C2	Bo th
C1	C2	C1	C2	C1	C2	C2	C1	C2	C1	C2	C1	C1	C2	C1	C1	C2	C1	C2	C1	C2	C1	C2	C1
C2	C1	C2	C1	C2	C1	C1	C2	C1	C1	C2	C2	C1	C1	C2	C1	C2	C2	C2	C1	C2	C2	C2	C2
Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	C2	C2	C2	Bo th	C2	Bo th	C2	C2	Bo th	Bo th	C1	Bo th	Bo th	Bo th	C2	Bo th	C2	C2
C2	C1	C2	C1	C2	C1	Bo th	Bo th	Bo th	C2	C1	C2	C1	C1	C1	C2	Bo th	Bo th	C1	C2	Bo th	C1	C2	Bo th
C1	C2	C1	C1	C2	C1	Bo th	C1	C2	C2	C1	C2	C2	C2	Bo th	C2	C2	Bo th	C1	C2	C1	C2	C1	Bo th
C2	Bo th	C2	C2	Bo th	C2	C2	Bo th	C2	C2	Bo th	C2	C2	Bo th	C2	C2	Bo th	C2	C2	Bo th	C2	C2	Bo th	C2
Bo th	Bo th	Bo th	C1	Bo th	Bo th	C2	Bo th	Bo th	C2	Bo th	Bo th	C1	Bo th	Bo th	Bo th	Bo th	Bo th	C1	Bo th	Bo th	C2	Bo th	Bo th
Bo th	C1	C2	C1	C2	C1	C1	Bo th	C1	C2	C1	C2	C2	C1	Bo th	C2	C1	C1	C1	C2	C1	C1	Bo th	C1
C2	C2	Bo th	C1	C2	Bo th	C1	C2	Bo th	C2	C1	Bo th	C1	C1	Bo th	C2	C2	Bo th	C2	C1	Bo th	C1	C2	Bo th
Bo th	C1	C2	Bo th	C1	C2	Bo th	C2	C1	Bo th	C1	C2	Bo th	C2	C1	Bo th	C1	C2	Bo th	C2	C2	Bo th	C1	C2
C2	C2	C2	C1	Bo th	C1	C2	Bo th	C2	C2	Bo th	C2	C1	Bo th	C1	C2	C2	C2	C1	C1	Bo th	Bo th	Bo th	Bo th
C1	C2	C1	C1	C2	C1	C2	C1	C2	C2	C1	C2	C2	C1	C2	C1	C2	C2	C1	C2	C1	C2	C1	C2

C1	C2	C1	C1	C2	Bo th	C1	C2	Bo th	C1	C2	C1	C2	C1	Bo th	C2	C1	C1	C1	C1	C1	C1	C2	C2
C2	C1	Bo th	C1	C2	C1	Bo th	C1	Bo th	C2	C1	Bo th	Bo th	Bo th	C1	C2	C1	Bo th	C1	C2	Bo th	C2	C1	Bo th
C1	C2	Bo th	C1	C1	Bo th	C2	C1	Bo th	C2	C1	Bo th	C1	C1	Bo th	C2	C1	Bo th	C1	C2	Bo th	C2	C1	Bo th
C1	C2	Bo th	Bo th	Bo th	Bo th	C2	C1	Bo th	C2	C1	C2	C1	C2	C1	Bo th	C1	Bo th	C1	C2	C1	C1	C2	Bo th
C1	C2	C1	C2	C1	C1	C1	C1	C1	C1	C2	C2	C1	C2	C2	C2	C1	C1	C2	C2	C1	C1	C2	C2
C1	C1	C2	C2	C1	C2	C2	C2	C2	C2	C2	C2	C1	C2	C2	C2	C2	C2	C1	C2	C1	C1	C2	C1
C1	C2	C1	C1	C2	C1	C2	C1	C2	C2	C1	C2	C1	C2	C1	C1	C2	C1	C1	C2	C1	C2	C1	C2
C2	C1	C2	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	C1	C1	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th
Bo th	C2	C2	C1	C1	C1	C2	C2	C2	C1	C1	C1	C1	C1	C1	C1	C1	C1	C1	C2	C2	C2	C1	C2
C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1
Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	C2	C2	C1	C2	C1	C2	C1	C2	C1	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	C2	C1	C2
Bo th	Bo th	Bo th	Bo th	C2	C1	C1	C1	C1	C2	C2	C2	Bo th	Bo th	Bo th	C2	C2	C2	Bo th	Bo th	C1	C1	C1	C1
C1	C2	Bo th	C1	C2	C1	C2	C1	Bo th	C2	C1	C2	C2	C1	C1	C1	C2	Bo th	C1	C2	Bo th	C2	C1	C2
C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C1	C2	C1	C2	C1	C2	C2	C1	C2	C1	C2	C1
C2	C1	C2	C1	C2	C1	Bo th	Bo th	C2	C2	C1	C2	C1	C2	C1	C2	C1	C2	C1	C2	C2	C2	C1	C2
C1	C2	Bo th	C1	C2	Bo th	C2	C1	Bo th	C1	C2	Bo th	C2	C1	Bo th	C2	C1	Bo th	C1	C2	Bo th	C2	C1	Bo th
Bo th	C2	Bo th	C2	Bo th	C2	Bo th	C2	Bo th	C2	Bo th	Bo th	C1	C2	C2	Bo th	C2	C2	C2	Bo th	Bo th	C1	C2	C2
Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	Bo th	C2	Bo th	C2	C2	C1	C2	C2	C2	C1	C2	C1	Bo th	C1	Bo th	Bo th

Questionnaire mapped table

Q	COM.	type
Q1	C1	new.
	C2	ORG.
Q2	C1	new.

	C2	ORG.
Q3	C1	ORG.
	C2	new.
Q4	C1	new.
	C2	ORG.
Q5	C1	ORG.
	C2	new.
Q6	C1	new.
	C2	ORG.
Q7	C1	ORG.
	C2	new.
Q8	C1	ORG.
	C2	new.
Q9	C1	ORG.
	C2	new.
Q10	C1	new.
	C2	ORG.
Q11	C1	new.
	C2	ORG.
Q12	C1	ORG.
	C2	new.
Q13	C1	org.
	C2	new.
Q14	C1	org.
	C2	new.
Q15	C1	new.
	C2	org.

Students Questionnaire first part:

very good	very good	Fair	3	Yes	Java	No	No	Yes
good	Fair	Fair	4	Yes	Java	No	yes	Yes
good	good	good	3	Yes	Java	No	yes	Yes
very good	good	good	4	Yes	Java	yes	yes	Yes
good	good	good	4	Yes	Java	No	No	Yes
good	good	good	4	No	Java	No	No	Yes
very good	good	good	5	Yes	Java	yes	yes	Yes
very good	good	very good		No	Java	yes	yes	Yes
good	good	good	4	Yes	Java	yes	yes	Yes
good	good	good	3	Yes	Java	yes	yes	Yes
very good	good	good	3	Yes	Java	yes	yes	Yes
Fair	Fair	Fair	3	Yes	Java	yes	yes	Yes
very good	good	good	5	Yes	php	yes	yes	Yes
good	good	Fair	3	Yes	Java	yes	yes	Yes
very good	very good	good	5	No	C, C++	yes	yes	Yes
very good	very good	good	5	Yes	.net (c#, Vb)	yes	yes	Yes
very good	good	good	3	Yes	C, C++	No	yes	Yes
good	good	good	2	No	C, C++	yes	yes	Yes
good	good	good	6	Yes	C, C++	yes	yes	Yes

[illegible]

org	ne w	org	Bot h	Bot h	Bot h				org	ne w	Bot h	Bot h	Bot h	Bot h				Bot h	org	Bot h
org	Bot h	org	Bot h	Bot h	Bot h	org	ne w	org	org	ne w	org	Bot h	Bot h	Bot h	org	ne w	org	org	ne w	Bot h
org	ne w	Bot h	ne w	org	Bot h	org	ne w	Bot h	org	org	Bot h	ne w	org	Bot h	org	org	Bot h	org	ne w	Bot h
org	ne w	org	ne w	org	ne w	org	ne w	org	ne w	org	org	org	ne w	ne w	org	ne w	org	org	ne w	org
ne w	org	Bot h	org	ne w	Bot h	org	ne w	Bot h	ne w	org	Bot h	ne w	org	Bot h	ne w	org	Bot h	org	ne w	Bot h
org	Bot h	org	ne w	ne w	Bot h	org	org	ne w	org	ne w	Bot h	org	ne w	ne w	ne w	ne w	Bot h	Bot h	Bot h	ne w
org	ne w	org	ne w	ne w	org	ne w	ne w	ne w	ne w	ne w	ne w	ne w	org	org	org	ne w	org	ne w	org	org
org	ne w	Bot h	ne w	org	Bot h	Bot h	Bot h	Bot h	ne w	org	ne w	org	ne w	org	org	ne w	org	ne w	org	Bot h
ne w	org	Bot h	ne w	org	ne w	Bot h	ne w	Bot h	ne w	org	ne w	Bot h	org	Bot h	org	org	Bot h	org	ne w	Bot h
org	ne w	Bot h	ne w	ne w	Bot h	ne w	ne w	ne w	org	org	org	ne w	ne w	ne w	org	org	org	ne w	ne w	ne w
ne w	org	org	ne w	org	ne w	org	ne w	org	org	ne w	org	org	ne w	org	org	ne w	org	org	ne w	org
org	ne w	org	ne w	org	ne w	org	ne w		Bot h	ne w	Bot h	org	Bot h	org	org	ne w	org	ne w	Bot h	ne w
org	ne w	org	ne w	org	ne w	org	ne w	org	Bot h	Bot h	Bot h	ne w	org	ne w	ne w	org	ne w	org	ne w	org
org	ne w	org	ne w	org	org	org	org	org	org	ne w	Bot h	Bot h	Bot h	Bot h	org	ne w	org	org	org	ne w
Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h	Bot h
org	ne w	org	org	ne w	org	org	ne w	org	org	ne w	org	org	ne w	org	org	ne w	org	org	ne w	org
org	ne w	org	org	ne w	Bot h	org	ne w	ne w	ne w	Bot h	Bot h	org	ne w	Bot h		Bot h	Bot h	ne w	org	Bot h
org	ne w	org	ne w	org	ne w	org	ne w	org	org	ne w	org	Bot h	Bot h	Bot h	ne w	org	ne w	Bot h	Bot h	Bot h
org	ne w	org	ne w	org	ne w	org	ne w	org	org	ne w	org	org	ne w	org	org	ne w	org	Bot h	Bot h	Bot h
org	org	org	org	org	Bot h	org	Bot h	Bot h	org	ne w	Bot h	org	org	Bot h	ne w	ne w	org	org	ne w	Bot h
org	ne w	Bot h	ne w	org	ne w	Bot h	Bot h	Bot h	org	ne w	org	Bot h	ne w	org	org	ne w	Bot h	org	ne w	org
ne w	ne w	ne w	ne w	org	ne w	org	ne w	org	ne w	ne w	ne w	org	org	org	ne w	ne w	ne w	org	org	org

Students Questionnaire part 2.2:

[illegible]

new	org	Both	new	org	new	org	new	Both	org	new	org	org	new	new	Both	Both	Both	org	new	Both	new	org	Both
org	new	Both	org	new	Both	org	new	Both	org	new	Both	new	org	Both	new	org	Both	org	new	Both	org	new	Both
org	new	org	org	new	org	new	org	new	org	new	org	org	new	org	org	new	org	org	new	org	org	new	org
org	new	Both	org	new	Both	Both	org	new	Both	org	new	Both	new	org	new	org	new	new	org	org	new	org	Both
new	org	org	Both	Both	org	Both	new	Both	org	new	org	org	new	Both	Both	new	org	org	new	new	new	new	org
new	new	new	org	new	org	new	org	new	new	org	new	org	new	org	org	org	org	new	new	new	new	org	new
org	new	org	org	new	Both	Both	org	new	Both	org	new	org	new	Both	org	new	org	new	org	new	Both	new	org
org	new	Both	org	new	Both	new	org	new	new	org	new	new	org	Both	org	new	org	org	new	org	new	org	new
new	new	new	org	org	org	new	new	new	new	new	new	org	org	org	new	new	new	new	new	new	org	org	org
org	new	org	org	new	org	new	org	org	new	org	new	org	new	org	new	org	new	org	new	org	org	new	org
org	Both	org	org	Both	new	org	Both	org	org	org	new							org	org	new			
new	org	new	org	new	org	org	new	org	org	new	org	org	new	org	new	org	new	org	new	org	org	new	org
new	new	new	new	Both	new	org	new	Both	new	new	new	org	new	new	org	new	Both	Both	new	Both	org	org	new
Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both	Both
org	new	org	org	new	org	org	new	org	org	new	org	org	new	org	org	new	org	org	new	org	org	new	org
Both	new	Both		new		new	org	Both	new	org	Both	org	new	org	new	org	org	Both	new	org	org	new	Both
org	new	org	new	org	new	org	new	org	org	new	org	Both	Both	Both	new	org	new	Both	Both	Both	org	new	org
new	org	new	org	new	Both	new	org	new	new	org	new	new	org	new	new	Both	Both	Both	new	Both	new	org	Both
Both	new	new	Both	new	Both	org	Both	org	org	new	Both	org	new	Both	org	new	new	org	org	org	org	new	org
Both	Both	Both	org	new	new	org	org	Both	new	Both	Both	org	new	org	org	new	org	Both	Both	Both	Both	org	org
org	org	org	org	org	org	new	new	new	new	new	new	org	org	org	org	org	org	org	org	org	new	new	new