Name	: Juhari bin Ijam
Matric No	: Wek 990341
Supervisor	: Prof. Dr. Syed Malek Fakar Duani
Moderator	: Puan Norisma Idris
Project Title	: Component-based Stemming Engine for Malay Text

ABSTRACT

Word stemming is an important feature supported by present day indexing and search system. The idea is to improve recall by automatic handling of word ending by reducing the words to their word roots, at the time of indexing and searching. Various algorithms for stemming have been developed for the English and the other foreign languages, but it is still new for the Malay text. How ever most of them did not given any meaning of the development or application. This is because it cannot be reused for the other applications. These projects are studied and a new algorithm is being proposed to improve the performance of the stemming process. And the most importance of this project is to propose a new technology, which is using component based. With it, a lot of applications may derive from the component. It is because the main reason of using component base is it can be reusable. So that for those who like to build a system which is have a relationship to the IR or word stemming, not need to build it anymore for the stemming engine. The developer has just to use the component engine and get the output easily. How ever this project is proposed for a specific domain that will be covered for the generic Malay words.

ACKNOWLEDGEMENT

A huge bundle of sticks become easy to carry when everyone carries a stick along with them. Teamwork reduces the amount of task each member has to do. Therefore the author would like to wish the following people who had directly and indirectly contributed to make this thesis possible.

First of all the author, would like to thank the almighty God, for everything that has given and for his guidance and blessing. The author would like to thank his supervisor, Assc. Prof. Dr. Sye Malek Fakar Duani bin Syed Mustafa whose guidance, advice and supervision throughout the development of the project success. He provided the author with much needed aid and guidance on every aspect of the project. His help is deeply greatly appreciated. Also not forgetting the authors moderator, Cik Norisma Idris who has given critical comments in the pursue of improving of the system. The author would like to thank her for his invaluable ideas and comments.

And finally the author would like to thank, his fellow friends especially Salhana, Kamarul, Sahidan, Norid, Azidi, Azman and the rest of them who had stayed with him throughout all weather and hardship. The support and comments gained from them made this very project possible. With them the experience the author gained as an undergraduate will be cherished forever. Thank you.

CONTENTS

LIST OF TABLE & FIGURES

List of tables

List of figures

CHAPTER 1

INTRODUCTION

1.1 Project Objective

In developing a certain project, requires the particular project to under go the development life cycle. This means that the project has to go through several stages of development before it is ready for implementation. The first stage in the development life cycle is setting the objectives of the project. Among the objectives that are to be achieved with this development of the Component Based Stemming Engine for Malay Text are :-

- To improve the efficiency of the Information Retrieval of a Malay text.
- To be used in education field, where writing exam online, especially for the essay questions possible to be marked automatically by the system. This will definitely reduce the time taken by lecture or teacher to mark the papers manually.
- To provide a general stemming engine component and can be used to anybody to develop their own application and system which is related to information retrieval.
- Can be used as a global dictionary. This mean that the user can use this component engine to check whether the word is correct or not or existing in the real dictionary. For this reason, all possible root word should be inserted first into the system.

This proposes project will use a lot of variable for the suffixes and special words. From that, we can apply this component engine to be used for any languages since it is a text based. Developer has just to insert any possible morphology – suffixes and a special word from their language to the component. This is very strength objective and feature that differ to all word stemmer application in the world.

1.2 Project Scope

Another important cycle in the development life cycle is setting the scope of the project. Without a proper scope, the project that is being developed is bound for failure. This is because the scope sets the fundamental rules of the project. As for this project development it will be divided into four main modules.

- Word Stemming for Malay Text
    - Brief story about Word Stemming
    - Algorithm
    - Input/output
- Component Based Stemming Engine
    - Input from file
    - Optional output
    - Usability
- User Interface
    - Sample Applications
    - UIs
- Multilanguage
- Data Dictionary

1.3 Thesis Organization

Generally there are six chapters in this project proposal. Each chapter contains the information of the different phrases of the Stemming Engine Malay text.

Chapter 1: Project Overview

This chapter contains the overall project overview the description of the project includes the objectives, the project scope and project methodology that specifies the development stages of the project.

Chapter 2: Literature Review

In this chapter, the information regarding the approach that will be used in this project will be presented here. This includes the definition of Information Retrieval, how an Information retrieval system works, the approaches that used in the Information Retrieval, and the link of Information Retrieval to the Stemming Algorithm.

Chapter 3: Methodology

Explain the model that is used to develop this component-based. Chapter 3 also will cover on all the phases that are being followed during the development of the system.

Chapter4: System Analysis and Design

All the fact-finding techniques that have been used to in this project will be presented here, plus the functional requirement, the non-functional requirement of the new system will also be included. The analysis of the system-developing tool includes all the hardware and software will be stated in this chapter.

Also a study of the previous stemming algorithms will be presented here plus the new proposed stemming algorithm will be described here as well as the results of the experiments being carried out.

This chapter will also cover the system specifications that have been planned. Also The flow chart of the new proposed system, the flow chart of the new stemming algorithm will be presented here also.

Chapter5: System Implementation

This chapter states all the physical design processes that involve a lot of algorithm and new methods. It concerns on the system architecture, the outcome of the reports and the screen. This also chapter contains system coding methodologies.

Chapter6: System Testing

This chapter explains and tests whether the system is functioning as the users requirement and if it is fulfilling the required specification. In also shows the method of testing that has been used to test this stemming engine component-based. Also included is the list of bugs that have been detected during the whole project development and their solutions.

Chapter 7: System Evaluation

In this chapter discusses the problem encountered during the development process of this project with their solutions. Also included are the system strengths and the limitations of the system. All the feedback from the users is also being documented in this chapter. The recommendations for future system enhancement will be stated as well.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The growing amount of information available mainly in the Internet poses significant problems in storing, managing and especially retrieving information. A central issue is retrieving information, which is relevant to the user's needs. This is the reason of intensive search in such areas as data mining, information extraction, information retrieval is being done. There are several schemes, which includes standard information retrieval methods, which can be considered as a basis for the modern advanced information retrieval techniques. On the other hand it includes techniques, which are based on the Artificial intelligence methodology.

2.2 Information Retrieval

In data retrieval, we are normally looking for the exact match. In other words we are checking to see whether an item is absent in a file. In the information retrieval, exact match sometimes be of interest, but more generally we want to find those items, which partially match the request and then select from those, the best matching items. If we will compare it with different types of systems we can clearly see that in a database queries context, one seeks facts that satisfy a query, in question answering (also know as information extraction) one seek specific answers to specific questions. In information retrieval, one seek a document that is likely to contain the information that a user needs. Information Retrieval(IR) system are used for retrieving document that are concerned with needs. It is a quite vivid that the interference tends to be inductive, rather than deductive; finding documents likely to be relevant requires skilful guessing. The problem with information retrieval is that the process of locating relevant documents is inherently uncertain and is highly context dependent. Figure 1 shows standard retrieval interaction model. It is evident, that the information retrieval is iterative process, which besides input and output operation includes the iterative reformulating. This activity changes query according to some certain strategy in order to increase relevancy of the received document.

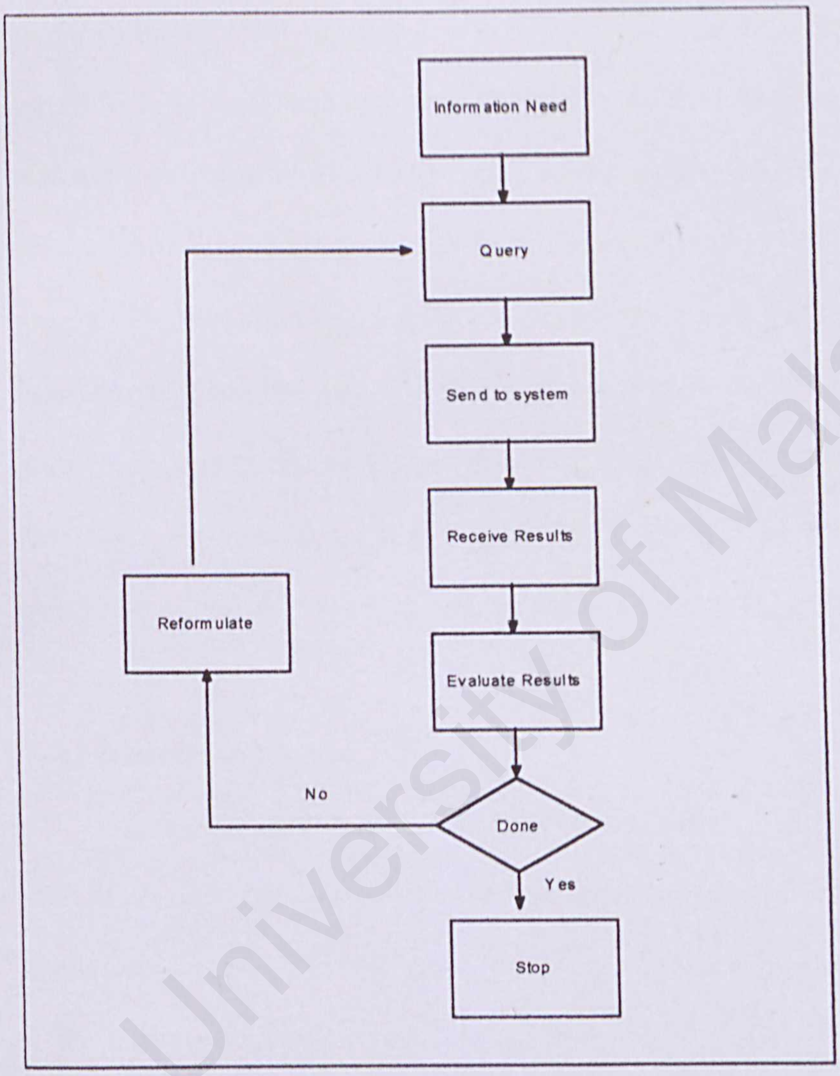


Figure 2.1 Standard Information Retrieval Interaction Model

It must be admitted, that information retrieval can be defined for any type of information such as text, images, video and sound. However, in this paper we will look through only text retrieval, because of it simplicity in methodology and implementation. It can be emphasized that described methods can be applied for example to multimedia retrieval too. All methods could be divided on two main parts that are *standard method* and *AI-based method.* The first group includes methods based on traditional mathematical or algorithmic techniques. The second tries to incorporate knowledge to the retrieval process by using artificial intelligence techniques to achieve better relevancy of the document.

2.3 Approaches to component technology

2.3.1 Introduction

The singe largest challenge facing software developers today is how they can undertake the transition of their applications (written in procedural languages) to a paradigm based on technologies such as object orientation and component-based assembly. Procedural languages were designed in the day when computer architecture was much simpler with 'green screen' terminals connected to a legacy application with character-based interfaces. The distributed applications of today require a different approach. Software reuse can be achieved through component-based development, and indeed the use of this technology today will facilitate software reuse in the future. Component-based development therefore assumes that developers will create an application by plugging together components, which have already been created, possibly by someone else in a different organization.

2.3.2 What is Component

Numerous definition exist for components. The term component has been used extensively, but rarely have proper design definitions been formulated. Visual Basic Extensions (VBXs) was one of the first to introduce the idea of components in 1992. Today's descriptions have evolved from this original idea.

There appears to be little general consensus on the exact definition of a component. It is therefore useful to look at the definitions offered by key researchers and practitioners in the area :

1. A business component represents the software implementation of an 'autonomous' business concept or business process. It consists of the

software artifacts necessary to express, implement and deploy the concepts as a reusable element of a larger business system [Wojtek Kozaczvnski[4]].

2. A component is a software module that publishes its interfaces [Paul Harmon, Cutter Information Corp[7]].
3. A component is a nontrivial nearly independent and replaceable part of a system that fulfils a clear function in the context of a well-defined architecture. A component conforms to and provide the physical realization of a set of interfaces [Philippe Krutchen, Rational Software[5]].
4. A software component is a unit of composition with contractually specified interfaces and explicit dependency only. A software component can be deployed independently and is subject to third-party composition [Clemens Szyperski, Component Software[6]] .
5. A runtime software component is a dynamic bindable of one or more programs managed as a unit and accessed through documented interfaces that can be made known at runtime [Gartner Group[8]].

The first expression of what a component is, point 1, sits at the top of the pyramid as business logic : points 2-4 are interested in the interface and point 5 is more low level as runable code definition *(see Figure 2.2)* [M.E.C. Hull, Computing & Control[12]]

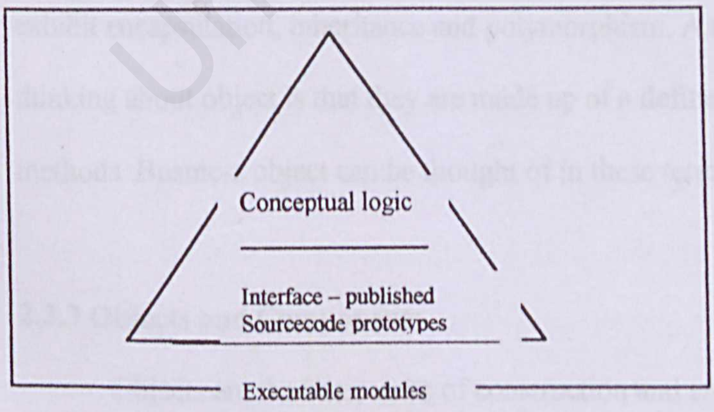


Figure 2.2 Various interpretations of where a component could be situated in a hierarchy

Paul Harmon[2] defines the overall goal of a component as the following:

" Component systems are clearly designed to facilitate reuse of code. Component-based development assumes that the developer will create an application by 'wiring' together components that have already been created by someone else. It also assumes that the person who created the components in the first place might not know how the components would be used. Thus, components are modules of code that are created with the intention that other developers will plug them into new applications."

Various attempts have been made at comparing components to objects. Szyperski[6] states that, in order to understand the difference between a component and an object, it is best to think of a component in terms of a class (or template or prototype). A component therefore is defined as the code that is written and always stays static, i.e. a component is developed and the code thereafter remains unchanged. An object (in this definition) can be considered as an instantiation of a component.

Riverton Software[6] provides a definition that relates more closely to software reuse and business objects. Objects are data structures that represent 'things' in the real world. The formal way to define them is that they are software entities that exhibit encapsulation, inheritance and polymorphism. A more practical way of thinking about object is that they are made up of a defined set of data and a set of methods. Business object can be thought of in these terms.

2.3.3 Objects and Components

Objects are the basic units of construction and are based on real-world entities of the application domain. Class templates are organized into libraries with common features, which are comprised of attributes and procedures called operations or methods. Objects communicate by message passing. Messages are often implemented as function calls. An object is used to hide the data within each instance.

Components on the other hand also have attributes associated with them, but are composed of two types of design constituent – the component and the component interface[6]. This facility allows the same component to be used in different contexts (see Figure 2.3).

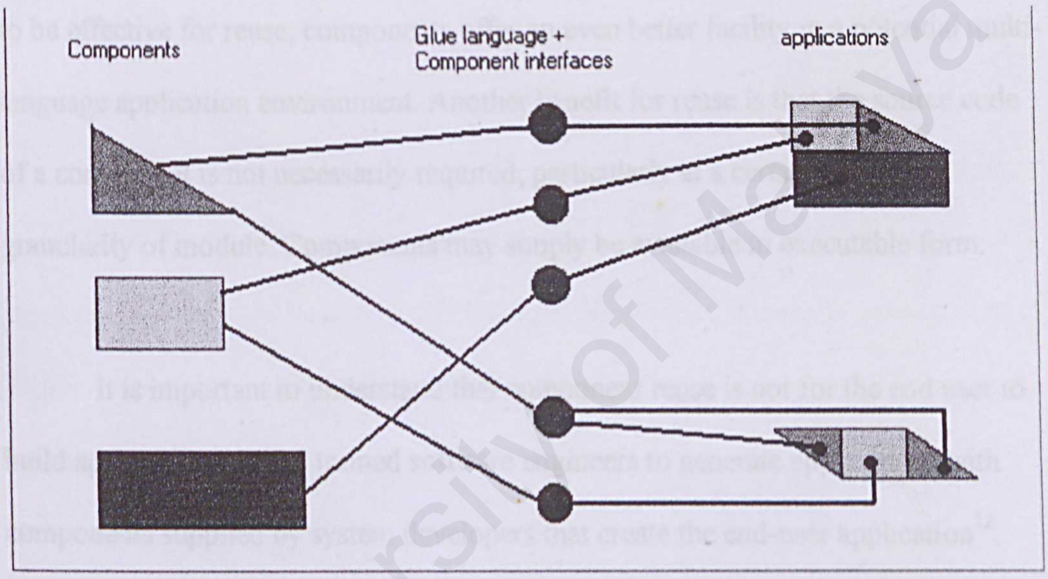


Figure 2.3 Component diagram

For component interact correctly, it is necessary to define the interfaces between components such that the interacting components understand what each must provide to the other hand also what s required by another component. For this reason it is the best that the interface is defined independently of each component, thus allowing the interface to be used in many different contexts, and hence the interface itself is de-coupled from each specific component, as shown in the figure.

This could mean that the data that was hidden within an object might be released to be accessed by other components.

It is interesting to note that components retain many of the benefit of object technology and at the same time do not have some of the 'difficult to use' aspects of inheritance and polymorphism, particularly when concerned with reuse. While it was always felt that object technology would deliver on reuse, that has been far from the case, but component technology has changed that. Where class library have proved to be effective for reuse, components offer an even better facility in a potential multi-language application environment. Another benefit for reuse is that the source code of a component is not necessarily required, particularly at a certain level of granularity of module. Components may simply be available in executable form.

It is important to understand that component reuse is not for the end user to build application, but for trained software engineers to generate applications with components supplied by system developers that create the end-user application[12].

In a client/server application there is a major benefit in the client-side component not being available in source code – this benefit the system software vendor and there in no real problem for the application developer either. However, from the server-side the situation is less clear as application developers may wish to customize components for their own business needs. This may actually be an advantage as it ensures that programmers using components cannot modify the code, which means that whatever business rules a component implements are properly enforced [Chappel[10]].

Component then are reusable, small modules. In object-based application, components will contain objects. Objects are written in an OO (object oriented) fashion whereas components may contain code written in OO, but might also be procedural or even written in assembly language. Current distributed architectures that are object-based use component as an abstraction or simplification mechanism. Components, which contain object, can abstract larger pieces of business (or computing) function so that the user workers at the level of the component – not the individual object.

2.3.4 Architectural overview of component systems

This section provides a study of component systems. Given that components are software modules that publish their interface it is important to understand how these interfaces will be defined and what kind of component systems are available to facilitate interface communication between different components. One of the leading component system is COM.

2.3.5 COM [Plassil[11]]

This is a component system for Microsoft Windows and consists of two key elements.

- COM interface
- A facility for passing messages between COM interfaces.

COM interfaces are encoded in MILD (Microsoft Interface Definition Language), but are first written by developers in C++ or VB and then compiled into an MIDL interface. A COM object is merely a collection of COM interfaces, but inside the interface it may contain pointers to other implementations of the interface (*Figure 2.4(a)*), contain the implementation of the interface (*Figure 2.4(b)*) or both.

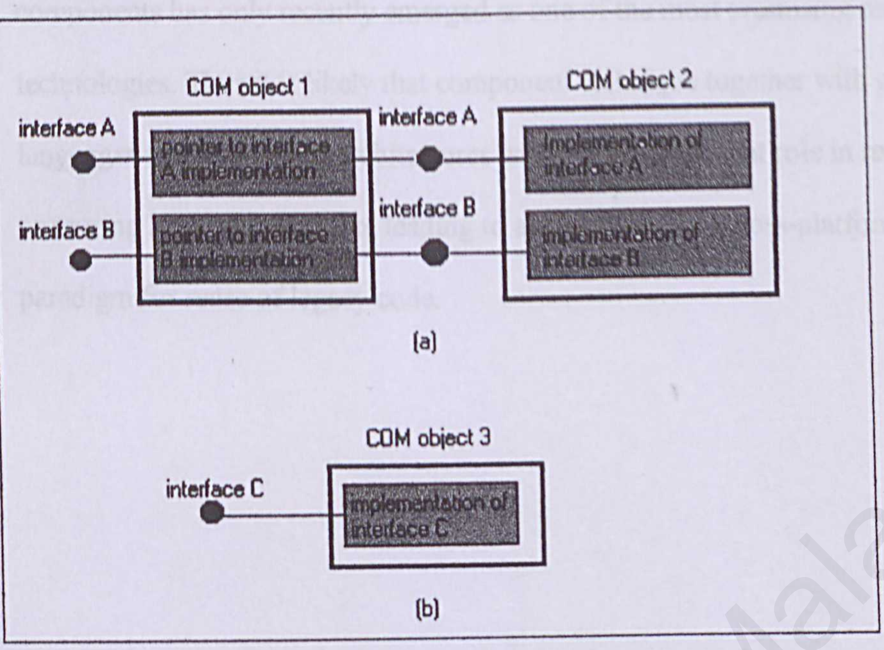

COM object 1
COM object 2
interface A
pointer to interface A implementation
interface A
Implementation of interface A
interface B
pointer to interface B implementation
interface B
implementation of interface B
(a)
COM object 3
interface C
implementation of interface C
(b)

Figure 2. 4(a) COM objects can contain pointers to COM implementation
Figure 2. 4(b) COM objects can contain interface implementation

All COM objects are compiled into a binary object so they are specific to the architecture on which they were compiled. Interfaces and components are both identified by globally unique identifiers (GUIDs). COM specifies how, given a pointer to a method, it can be called using specified parameters and in which order.

COM was originally developed as a component system for Microsoft Windows. With the development of Windows NT, Microsoft has extended COM to support cross-platform communication. DCOM is simply COM plus and ORB (Object Request Broker).

In this paper, we have investigated the characteristics of components, the difference between a component and an object, and their potential application in a distributed environment. Despite early calls for the approaches of component-based software development going back to the late 1960s, development based on software components has only recently emerged as one of the most promising reuse technologies. Thus it is likely that component technique together with glue types of languages and distributed architectures will play an important role in reconstructing or reusing legacy application, leading to across-language, cross-platform, new paradigm for reuse of legacy code.

CHAPTER 3

METHODOLOGY

3.1 Methodology Overview

Methodology is defined as a collection of procedure, techniques, tools and any pattern of documentation. In developing a system, a development model that consists of system development process phase must be shown to help users and the system developer. By doing this also, a system developer can make an early planning and evaluating on the activities that will be implemented during the development process.

The advantages in modelling the system development process are :-

1. Generate a simple understanding on implementation of activities, resources and limit allocation that might occur.
2. Modelling the implementation activities during the system development will help in achieving the most effective ways to counter any non-conjunction in any of the development phases.
3. By using a model in developing a system, the sequence of phase is related to the phases before and after it. This relation will help developer to allocate cost with the time given in every software development system phases.
4. By using a model in developing a system, the implementation of a particular process can be tracked.

3.2 Development Model

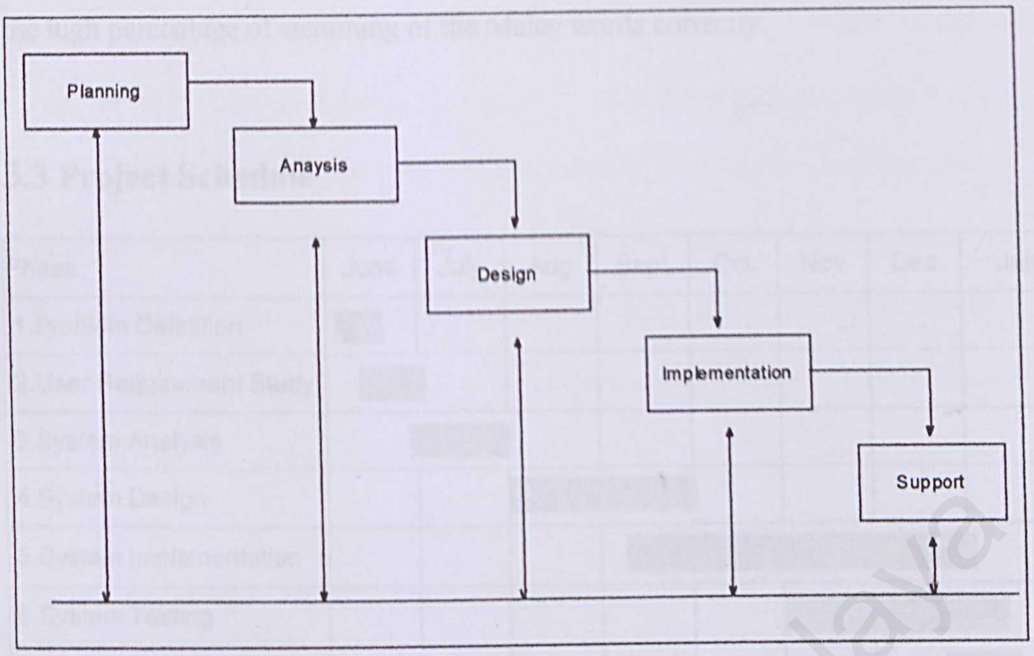


Figure 3.1 Waterfall Model

The project development methodology of Component-based Stemming Engine for Malay text is the Waterfall approach. The Waterfall model builds correction pathways into the model that enable a return to a previous phase. It is the most widely used methodology to implement the system development life cycle. As shown in the figure 3.1, the methodology consists of five phases including planning analysis design implementation and support.

In the planning phase, the current problem will be identified, the need of the project will be recognized and the project objectives will be set. The analysis phase involves the processes of analysis the existing methods of the stemming algorithm, the analysis of the approaches used and etc. After system analysis will be the system design. The design phase concerns on the system architecture, flow charts and system specifications .The system program design is followed by the system implementation where the program will be developed and tested for execution. In the final phase, the new system program will be ensured that it has met its goal, which is the high percentage of stemming of the Malay words correctly.

3.3 Project Schedule

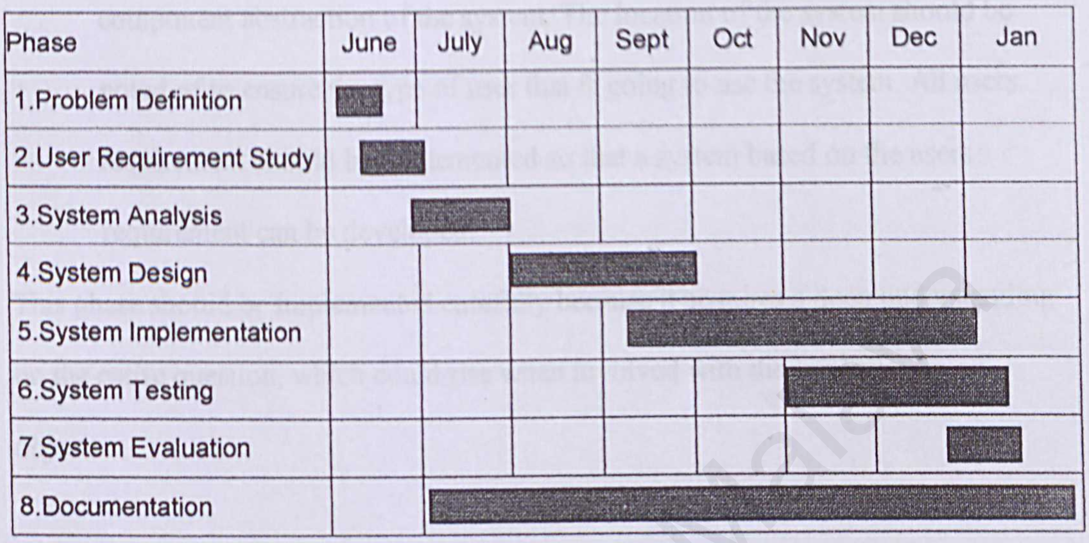


Figure 3.2 Project Schedule of The Word Stemmer Application

Above is the project schedule that has been followed as guidance for the development of this Word Stemmer Application.

3.3.1 Requirement Analysis Phase

Involved initial research activity, literature review, system component analysis, and problem in developing the system. There are three main activities :-

a) Initial research - Involved the main reason the system is being build, system definition, scope and objective to be reached in developing the system and planning on certain activity implementation throughout the development process.

b) Analysis research - Concentrate on the system requirement. This phase covers the searching and data and information analysis that are related to detect any problems and system requirement. Strategy and planning should be arranged to collect the necessary data and information. This can be seen from the component abstraction of the system. The location of the system should be noted of to ensure the type of user that is going to use the system. All users requirement should be implemented so that a system based on the users requirement can be developed.

This phase should be implemented carefully because it acquires a deep understanding on the entire question, which could rise when involved with the implementation object.

3.3.2 System Design Phase

This phase focuses on the process of developing the system that covers the activities such as :-

- Illustrating the system model architecture
- Designing the graphical user interface
- Determine the model that will be used to develop the system
- Illustrating the concept design and the system's technical design

3.3.3 Programming Design Phase

This phase focuses on the system technical design that has been grated on the system design phase to the programming design phase. Technical visualization is implemented according to the flow chart descriptions that will be the model during the process of coding the flow to the coding process.

3.3.4 Coding Phase

In this phase, the coding process is implemented. The flow chart, which has been created earlier, will be transformed into a code form. This phase is important because it serves as the backbone in a system. Coding must be done accurately to produce a coding shape with a high quality codes. An exceptional handling strategy should be applied because it will help in maintaining problem due to programming codes.

3.3.5 Unit Testing Phase

The testing is conducted by using a realistic data. A system's user are also involved to determine the function that user would want to use. This phase will use certain technique on certain function model that is built in the system.

3.3.6 System Testing Phase

This phase involved integration between the modules that is builds in the system. In this phase, all modules will be combined so that it can function as a complete system and will be tested as the whole system. This testing will confirm whether the system objective is archived or not by referring to the validation and verification process. Testing that will be carried out during the system testing process are such as functional testing, ability testing and installation testing.

3.3.7 Operational and Maintenance Phase

This phase will provide the documentation on how to use the system with the steps underlined by the developer. This documentation also will include all the description on handling problems if there is a problem that occurs during the operation of the system. Developers provide this documentation to ensure that the user will be able to adapt to a correct usage of the system. This documentation also can help a user to face a certain situation without need to refer to the system's developer.

3.4 Fact-finding Techniques

Fact-finding technique is a way to find related items, knowledge or data for the system that is going to be built. Listed here are some of the techniques used for developing the component-based for stemming engine.

3.4.1 Observations

In this technique, the system developer observes how the operations/task are performed. The techniques allows the developer to gather information about the system, the people who are involved, what, when, where and why. This technique can be used to verify some of the facts gathered through other techniques.

3.4.1.1 Observation Advantages

- Helps to build relationships with the operating staff :-
- By actually seeing or participating in the system operation, it helps the developer to understand the system better and it might provide additional perspective about the current system.
- This is one of the most inexpensive technique, compare to others like interviewing and questionnaire.

3.4.1.2 Observation Disadvantages

Direct observation might distant the staff/users and may result in poorer performance. This will affect the result of the observation.
Sometimes, the staff might just let the developer see what they want the developer to see. This may not be an actual performance level. The developer may not be able to see some of the unusual or unexpected situations that occur only occasionally.

3.4.2 Review Of Documents

The system developer gets through the existing documents to understand the system and its operation. The developer may collect the relevant documents from the individual who attended the interview or actually use the system.

3.4.2.1 Review Of Documents Advantages

- By understanding the existing system, the developer will be able to. Its strength and weakness and thus help them to design a better system.
- It is economical.

3.4.2.2 Review Of Document Disadvantages

- Information on the document might be out of date or not available.
- The document procedure might have change or even eliminated.
- The document may be difficult to read and understand or may be complex.

CHAPTER 4

SYSTEM ANALYSIS AND DESIGN

4.1 System Analysis Overview

System analysis is an activity that seeks then systematically analyzes data input or data flow, processing or transformation of data, data storage and information output within the context of a particular business.

Thus the objective of this phase is to define a component-based stemming engine for Malay text that includes, those that are needed to fulfill the system's purposes. This phase will also explain how these functions work with other systems.

4.2 System Requirements

System requirement is the official statement of what is required of the system developers. It should set out what the system should do without specifying how it should be done. System requirements will be described in two ways: functional requirements and non-functional requirements.

4.2.1 Functional Requirement

Functional requirements describe an interaction between the system and its environment. For example, to determine the functional requirement, it is necessary to determine what are acceptable states for the system to be in. further, the functional requirements describe how the system should behave given certain stimuli. In short, functional requirements describe system services of function.

In principle, the functional requirement definition of a system should be both complete and consistent. Completeness means that all services required by the users should be define. Consistency means that requirement should not have contradictory definitions. The functional requirements also explicitly state what the system should not do. Below are the functional requirements of the stemming algorithm system.

4.2.1.1 Functionality

1. OOP Based – Can be used to every developer.
2. Advance searching – Searching engine for web-based / dictionary.
3. Education field – Essay grading.
    a. Can be used to many platform application;
        i. Web Based (ASP, JSP, PHP).
        ii. System (Visual C++, Visual Basic)

    b. Operating System (MS Windows, Unix/Linux)

    Why? Using Binary code after compiled to specific file (.dll,exe).

4.2.1.2 Input Functions

1. The system must accept the following inputs:-
    - Object declaration to archive dictionaries, and suffixes file. Malay files are a default files.
    - A single Malay word
    - Sentences of the Malay word
2. All the inputs are either key in by using the keyboard.
3. All inputted Malay words are bound to the Malay morphology rules.
4. Affixes such as Prefix, Suffix, Infix and also Prefix-Suffix will be accepted.
5. Document file in text format to be stemmed.

4.2.1.3 Output Functions

1. The system must generate the following statistical analysis outputs:
    - Stemmed words/sentence
    - Number of word occurrences in the text
    - Number of words stemmed
    - Number of words stemmed correctly
    - Number of words unchanged

2. The input can be viewed ate the screen or be manipulated for specific purpose.

4.2.1.4 Processing Functions

1. The system must perform the following processes according to the Stemming algorithm: -
    - Remove Prefix's
    - Remove Suffix's
    - Remove Infix's
    - Remove Prefix-Suffix's
    - Check the local and general dictionary
    - Replacement of certain exceptional.

4.2.1.5 Storage Functions

1. The system will not be involved in any storage activities.
2. All data's from the dictionary will be stored in a file.
3. Data's from the file can be created, amend, and deleted.

4. The results of the stemming procedure will not be saved in a database.
5. All prefixes, suffixes and infixes will be stored in the text file with a specific name of language file name (*eg.: prefixes-BM.txt, prefixes-EG-txt*).

4.3 Previous Work

There have been two previous researches being done in the process of developing the Malay words stemming algorithm. The first was in 1993 and it has been improved by Sembok in 1994. This section describes briefly the two algorithms and explains some problems in both algorithms.

4.3.1 Othman's Algorithm

Othman developed the algorithm in 1993. The algorithm used 121 rules, which defined prefixes, suffixes, infixes and prefix-suffix pairs, and in general, rules are defined as follows:

1. Prefix rules format: Prefix +

   e.g. di + jajah -> *dijajah*

2. Suffix rules format : +Suffix

   e.g. jajh + an -> *jajahan*

3. Infix rules format: + Infix+

   e.g. tapak (+ el +) -> *telapak*

4. Prefix-Suffix pair rule format : Prefix + Suffix

   e.g. menN + takluk -> *menakluki*

The basic algorithm of the stemmer has several steps:

Step 1: If there are no more words, then stop, otherwise set the next word

Step 2: If there are no more rues, then accept the word as root word and go to Step 1, otherwise get the next rule

Step 3:Check the given pattern of the rule with the word. If it matches, then apply the rule to the word to get a stem

Step 4:Check the stem against the dictionary. Perform any necessary recording and recheck the dictionary

Step 5:If the stem appears in the dictionary, then the stem is the root of the word and go to Step1 otherwise go to Step 2

The algorithm adopted a rule-based approach that slowed down the stemming process. The stemmer also did not take into consideration that the stemmers would remove affixes from the root words, because the dictionary is not checked until after the first rule has been applied to the word. For example, the word "tempatan"(local) would be stemmed to 'tempat' (place) where 'an' is consider a suffix. These two words have different meaning and may affect the performance of the information retrieval system.

4.3.2 Sembok's Algorithm

The algorithm is a modification version of Othman's algorithm where it adopted Othman's algorithm and a set of morphological rules as the basis for the development of two new rule sets. The first contained 432 rules of affixes and the second set contained 561 rules of affixes

In order to enhance the original stemmer developed by Othman, another step has been added to the basic algorithm of the stemmer where the input word is checked first against the dictionary at the initial step. The main purpose for the additional step is to avoid stemming on the word, which is already a root word, in which otherwise, could lead to over stemming. For example, the word 'mati' (die) exists in the dictionary. Therefore, the word is not stemmed to the word 'mat' even though it contains the suffix 'i'. The other experiment carried by Sembok was to determine the order of the rules. It is important to know that the algorithm of the stemmer relies on the order of the rules (which affixes has to be checked and removed first).

This algorithm has been tested to the Quran and research abstract data sets. The experiments showed a significant improvement in stemming the Malay text and have been guidance for building a new stemming algorithm.

4.4 The New Stemming Algorithm

With the main purpose of research requires us to improve the effectiveness in the retrieval process. The main interest is not only to develop a new stemming algorithm for a specific application, but also to enhance the performance of the existing algorithm in order to improve the retrieval process. There has been another stemming algorithm, which have been developed by Norisma in 2000. It is a modification of the both previous stemming algorithm module. Thus this algorithm developed by Norisma is less error prone, and more effective in stemming the Malay words. The only defect with this algorithm is that the Infix rules is abandoned due to its unpopularity of use in the formation of new Malay words. In this new system which have been developed by Somabalan, the infix rules will be included. So the Norisma's algorithm will be used, as it is plus some additional Steps to be included for the Infix rules. Below is Norisma's algorithm with the modification statements following after it.

4.4.1 Norisma's Algorithm

In this new algorithm, only the most important rules from the two patterns of the rules, which are Prefix and Suffix will be implemented. And as for the Prefix-Suffix pattern because, it is actually the combination of the Prefix and the Suffix, so there will not be a set of new rules for these pattern. By using only two patterns of affixes that are prefix and suffix, we can reduce the number of rules sets. (Refer Appendix A for the flow chart of Norisma's stemming algorithm).

Prefixes usually give rise to the spelling variations and exceptions in the root word. In this case, errors may occur during the stemming process where the stemmed word is not complete. To deal with this kind of problem, we build another rule, which we refer to as Rule 2 where it can be applied to the prefixes removed only. The rules of *Rule 2* are:

```
If the stemmed word is not complete, then
    Check the first letter of the word
    If the stemmed word started with the vowels letters then
        1. Add 't' after removing 'men' or 'pen'
        2. Add 'k' after removing 'meng' or 'peng'
```

3. *Add 's' after removing 'meny' or 'peny'*

4. *Add 'f' or 'p' after removing 'mem' or 'pem'*

For example after removing the prefix 'mem' from a word 'menakluk' (to conquer), a letter 't' would be attached to the remaining word 'akluk'to form the correct stemmed word, 'takluk'(conquer'. But, for the word 'mentadbir' (to manage), the stemmed word is 'tabir' (manage), so we do not have to apply this rule as the stemmed word started with the consonants letter 't'. Basically the algorithm is: -

Step1: Check the word against a general dictionary. If the word is found in the dictionary, then accept the word as the root word and exit, otherwise proceed to the next step.

Step 2:Check the word against a general dictionary. If the word is found in the dictionary, then accept the word as the root word and exit, otherwise proceed to the next step.

Step 3:Check the word against the prefix rules. If the word matches the prefix rules , check the pattern of the prefix and the first letter of the stem word, otherwise go to Step 8.

Step 4:If the pattern of the prefix matches the prefix patterns in Rule 2, then apply the Rule 2 to the words, otherwise remove the prefix and go to Step 7.

Step 5:Check the prefix of the word against the pattern of Rule 2. If it matches the fourth rule (Rule 2(4)), then check the new stem word against the dictionary and proceed to the next step, otherwise remove the prefix and go to Step 7.

Step 6:If the word is not found in the dictionary, then go back to Step 5, otherwise remove the prefix and proceed to the next step.

Step 7:Check the word against the dictionary. If the word found in the dictionary, then accept the word as the root word and exit, otherwise proceed to the next step.

Step 8: Check the word against the Suffix rules. If it matches with the Suffix rules, then remove the suffix and go to Step 1, Otherwise, just go to the Step 1.

4.6.2 Modification from Somabalan

There are two main modifications that are suggested. There are: -

Step 8:Check the word against the Suffix rules. If it matches with the Suffix rules, then remove the suffix and check the dictionary. If the word is found, then accept the word as the root word and exit, otherwise, If the Prefix have been removed then add again the prefix and go to Step 9 else go to Step 9.

Step 9:Check the word against the infix rules. If the word matches the Prefix rules, then remove the infix and go to Step 1. Otherwise if Prefix has been removed before and added again, now remove the prefix again and go to Step 1 else proceed to Step 1.

As you can see there have been some changes in Step 8. Plus another Step has been added to the algorithm that is Step 9. The reason for the changes in Step 8 is to make sure that if a word does not have any prefix or suffix attached, then the word should be checked with the infix rules. And the rule '*if the prefix has been removed before it arrived to Step 8 then add the prefix again*' before proceeding, is to ensure that the particular word is not wrongly stemmed. For example the word 'telapak', when it reaches Step 8, the prefix 'te' would have been removed in Step 4.The balance of the word 'lapak' does not make any sense. So to overcome this problem when the word reaches Step 8, and it finds out that there is no suffix attached to the word, and before proceeding to Step 9, the prefix that has been removed is added back to the word. So by doing this, in Step 9, if the particular word is a word with the infix rules, the search should come to an end. This is because an Infix word does not have any prefix or suffix attached to it. With the Addition of these new rules, the flow of Norisma's algorithm will not be interrupted as well for the efficiency of the algorithm.

There were several experiments carried out before we eventually came out with the modification of the stemming algorithm. The first experiments were to determine where the checking of the infix rules should be inserted in Norisma's algorithm. It is important to know that the algorithm of the stemmer relies on the order of the rules. By performing some sideshow experiments, it is found that the place to place the checking of the Infix rules is after Step 8. However, to our knowledge there are only 4 infix rules currently available in the formation of Malay words. The results of the experiments are reviewed in Table 4.1.

Word	Actual Root Word	Infix	Modified Stemming Algorithm
Kelelawar	Kelawar	el	Kelawar
Selerak	Serak	el	Serak
Telapak	Tapak	el	Tapak
Telekup	Tekup	el	Telekup
Gemuntur	Guntur	em	Guntur
Gemilang	Gilang	em	Gilang
Semerbak	Serbak	em	Serbak
Gerigis	Gigis	er	Gigis
Serabut	Sabut	er	Sabut
Lelaki	Laki	el	Lelaki
Seinambung	Seimbung	in	Sambung

Table 4.1: The Experiments Results On Infix Words

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 Introduction

This chapter will cover on the implementation of the system. Since this project is stressed more on the research, so this chapter will concentrate on implementing the stemming engine, which involved component-based.

5.2 Development Environment

Using a suitable hardware and software will not only help to speed up the system development but also determine the success of the project. The hardware and software tools used to develop the entire system are as follows.

5.2.1 Hardware Requirement

In developing the component-based stemming engine, the configuration of hardware that has been used is:

- Intel Pentium III Processor (733MHz)
- 15'' SVGA Color Monitor (32bits, 800x600)
- 128MB SDRAM
- 15 GB Hard Disk Drive
- 1.44MB Floppy Drive Driver
- 52x CD-Rom

5.2.2 Software Requirement

The required software to run the system and develop the system:

- Microsoft Windows 98
  - Stand as operating system that supports all software development tools.

- Visual C++ / Visual Studio
  - For application development, which is all algorithm, are implemented and is programmed to build the real system.
- Internet Explorer
  - To view the debugging messages.
- Microsoft notepad
  - Dictionary data storage for
    - Global dictionary
    - Local dictionary
    - Prefixes files
    - Suffixes files
    - Infixes files
- Mic. Visio Professional
  - To draw ER and DFD Diagrams.

5.3 System Development

Visual C++ is the main important tools in developed this system. It will involve classes, inheritance, constructors, destructors, operator overloading, objects, encapsulation and etc. This system is build as a Win32 Console Application and plus inheriting from the Microsoft Foundation Class(MFS). Every application development project needs its own project workspace in Visual C++. The workspace includes the directories where the application source code is kept, as well as the directories where the various build configuration files are located.

5.3.1 Module Coding

The main part in creating the component-based system is the reusable of the code. These mean that the code should be programmed with a lot of variable instead of hard-corded (put the data to the programs). So, every time the programs calling the data such as dictionary, suffixes, prefixes and infixes, it will invoke external file. The file is save in a txt format. This is a good practice of implementing a component-based system where the end user(normally developer) not need to know what is the process in the system. What they need to do is to declare their own object as a stemmer class and send required information. They also should know how to use component-based in programming observation. The end user can modify the external file to meet the requirements. These mean that they can add any root words in the dictionary, prefixes, suffixes and infixes in a specific files.

5.3.1.1 Class Definition

Below is the code segment for stemmer class definition with the method provided.

```
class stemmer{
private:
      CString fileGlobalDictionary, fileLocalDictionary;
      CString filePrefixes, fileSuffixes , fileInfixes;
      CString wordIn, rootword;
      CString sentenceIn;
      CString stemmedSentence;
      int totalStemmedWords;
      int totalWordsIn;
      int totalOutput;
public:
      stemmer(CString ="globalDic-BM.txt", CString ="localDic-
BM.txt");
      void testOutput();
      void stem(char *sentence,int caseSensetive=0,int tag=0);
      CString stemWord(CString word,int caseSensetive=0,int tag=0);
      int globalDictionary(CString word);
      CString localDictionary(CString word);
      CString Prefix(CString word);
      CString Suffix(CString word);
      CString Infix(CString word);
};
```

5.3.1.2 Method Usage

In this constructor function, `stemmer(CString ="globalDic-BM.txt", CString ="localDic-BM.txt", CString ="prefixes-BM.txt", CString ="suffixes-BM.txt", CString ="infixes-BM.txt");`, user have to sent the dictionary file path for both global and local as the first two attribute and the rest are the prefix file, suffix file and infix file. All the files should in the same language or morphology. This is important because the system/component will give wrong result if the file is wrong. For example if the dictionary file is `globalDic-BM.txt` which is present for Bahasa Melayu, so all the rest file should also in Bahasa Melayu for the contents. But, the component still run with the default language as Malay if this constructor is created without attribute such as `stemmer();`. This is because to make it less error while coding and the most important is to fulfill the requirement of the title Component-Based Stemming Engine for Malay Text.

With this approach users can use this component to stem other language as far as the language is a text based. It is also easy for user to put in the file if they got a new words or suffixes. It can be implemented in a better user interface input such as open a new language in a file menu.

```
void stem(char *sentence,int caseSensetive=0,int tag=0);
```

With the function above user can stem their sentence with their own option. The first attribute is the sentence address, the second attribute for the case sensitivity. 0 for the default (not case sensitive) and 1 for case sensitive. The third attribute is for tagging option.

```
0: Default, stem symbol n not stem tagging
1: stem symbol only
```

```
2: stem tagging and symbol


int globalDictionary(CString word);

CString localDictionary(CString word);

CString Prefix(CString word);

CString Suffix(CString word);

CString Infix(CString word);
```

The three functions/components above is used for internal program to find the root word and make replacement if any.

Function	Output Returned
`CString sentenceIn;`	Original sentences from input.
`CString stemmedSentence;`	Stemmed sentences with a specific language.
`int totalStemmedWords;`	Total stemmed words, which is involved in the stripping for prefixes, suffixes and infixes.
`int totalWordsIn;`	Total words from input.
`int totalOutput;`	Total words will appear after stemming process.

Table 5.1: Output for each function.

The functions/components above can be used to get the output by the end user. Sample code for using the components is shown in Appendix F.

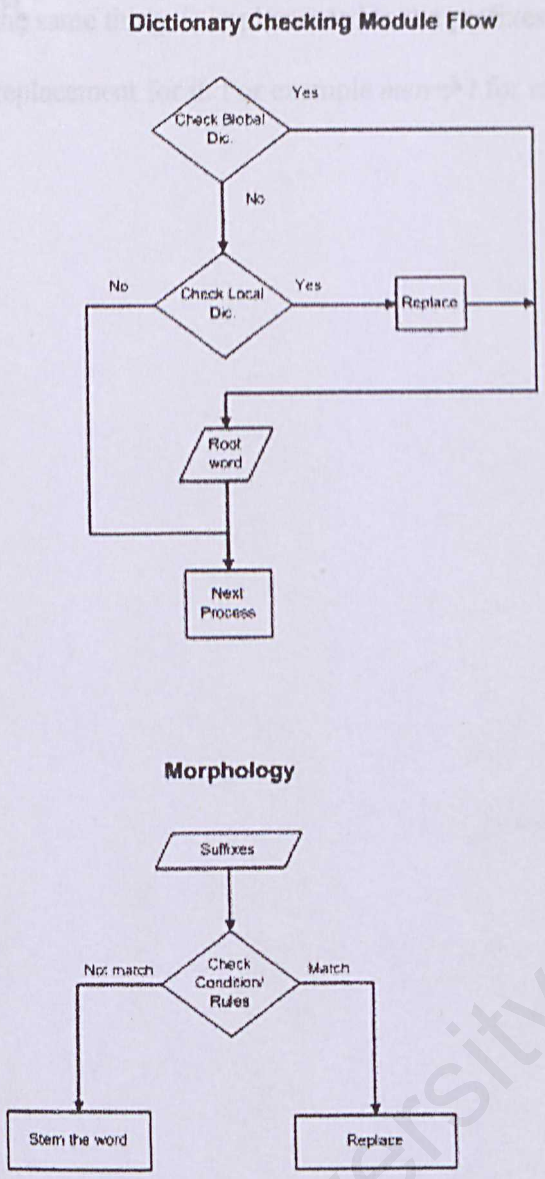


Figure 5.1: Dictionary and morphology replacement module

Figure 5.1 shows a dictionary and morphology module with special replacement, which implemented as the code in the Appendix F and using new diagram from Appendix C. The word will be sent to dictionary module as stemmed or unstem word. It will find a match word in the global dictionary with the input word. If matches, then it will return as root word, if not it will try to find in the local dictionary. If the word found, it will use the replacement word as root word else it will not return as root word. With this approach the problem for special words for their root words are solved. For example in English; *ran* → *run, fought*→*fight*. So, the same thing, is implemented to the prefixes or suffixes which will make replacement for it. For example *men*→*t* for **men***akluk*→**t***akluk*, **men***tadbir*→**t***adbir*.

CHAPTER 6

SYSTEM TESTING

6.1 Introduction

System testing is a crucial process in developing any software. This is a process where the system will be verified and validated in term of the system functional requirement, performance, reliability and specifications. However the stemming component will only be tested on the functional requirement.

The objective of unit and integration testing is to ensure that the code implemented the design properly; that the programmers wrote code to do what the designers intended. In system testing, the objective is different: to ensure that the system does what the customer wants it to do. To understand how to meet this objective, first we must determine where faults in system come from

The most widely used testing process consists of five stages shown in Figure 6.1 below:

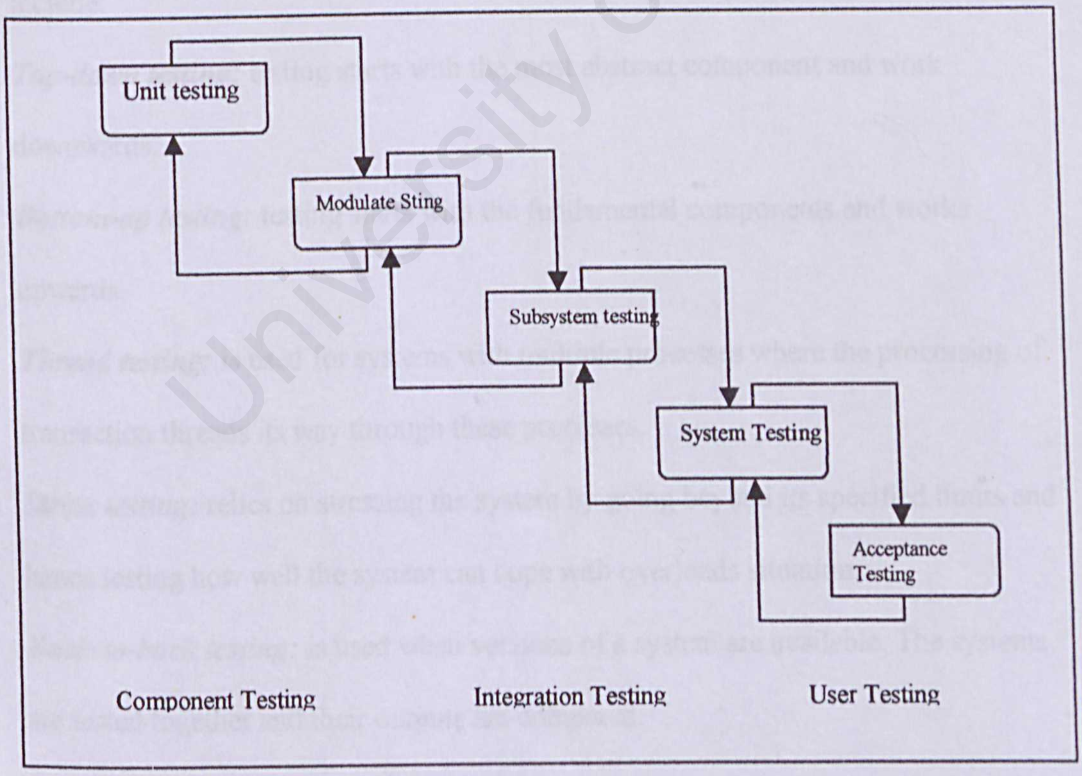


Figure 6.1 Testing Process Stages

The sequence of testing activities is component testing, integration testing then user testing. As defects are discovered at any stage, program modification are required to correct them and this may require other stages in the testing process to be repeated. The process is therefore an interactive one with information being fed back from later stages to earlier parts of the process.

In Figure 6.1, the arrows from the top of the boxes indicate the normal sequence of testing. The arrows returning to the previous box indicate that previous testing stages may have to be repeated.

6.1.1 Testing Strategies

A testing strategy is a general approach to the testing process rather than a method of devising particular system or component tests. The testing strategies include:

Top-down testing: testing starts with the most abstract component and work downwards.

Bottom-up testing: testing starts with the fundamental components and works upwards.

Thread testing: is used for systems with multiple processes where the processing of transaction threads its way through these processes.

Stress testing: relies on stressing the system by going beyond its specified limits and hence testing how well the system can cope with overloads situations.

Back-to-back testing: is used when versions of a system are available. The systems are tested together and their outputs are compared.

6.2 Unit Testing

Unit testing verifies that the component functions properly with the types of input expected from studying the component's design. The first step is to examine the program code by reading through it, trying to spot algorithm, data and syntax faults.

All algorithms for every module are tested with appropriate input. The modules are constructor component, dictionary component, morphology component (prefix, suffix and infix) and the user output component.

Below are the testing results;

Unit Testing	**Result**
Constructor component	Worked properly. The default value was taken if the constructor created without attribute. Eg: stemmer obj1; Or stemmer obj1("globalDic-BM", "localDic-BM");
Global dictionary component	Worked properly. The global dictionary returned the true root word. e.g.: *adik* → *TRUE* *adun* → *TRUE*

	akhbar → *TRUE* *pita* → *TRUE* *polis* → *TRUE* *longan(not in the dictionary)* → *FALSE* *pistol(not in the dictionary)* → *FALSE* The testing return FALSE because the words are not existed in the global dictionary.
Local dictionary component	Worked properly. Test for Malay text; *perkosa*→ *rogol* *sebuah*→*buah* How ever it is too few for Malay language. This component is very useful for other language such as English where there are so many exceptional words. Test for English text; *ran*→*run* *bought*→*buy* *men*→*man* *broke*→*break* *drunk*→*drink*
Prefixes component	Worked properly with some modification. Data tested for Malay words;

	memberi→beri *dilantik→lantik* *berkata→kata* *bersedia→sedia* Problem occurred when overlapped of these prefix *meng*, *memper men, menye* and *me*. And combination of pe, peng, penye Eg; *mencari→ncari (strip the me)* *mengguna→ngguna, gguna* *pengguna→ngguna* *penyapu→nyapu* This problem occurred because of the improper prefix list (unsorted prefix). To solve this problem the prefixes list are listed properly with the highest length of prefix at the top. Suffixes and infixes files also made with the same changes.
Suffixes component	Worked properly after changes *racuni→racun* *perlukan→perlu* *mulai→mula* *tempatan→tempat* Combination of prefix-suffix *perlindungan→lindung*

	perkataan→kata
	permulaan→mula
	menyeksakan→seksa
	mempercayai→percaya
Infixes component	Worked properly after changes
	telapak→tapak
	gemuruh→guruh
	gerigi→gigi
	jejari→jari
	sinambung→sambung
`CString sentenceIn;`	Sentence tested:
	Suatu proses ialah beberapa langkah yang melibatkan aktiviti, kekangan dan sumber yang akan menghasilkan output yang diingini <lulus>.
`CString stemmedSentence;`	*Suatu proses ialah berapa langkah yang libat aktiviti kekangan dan sumber yang akan hasil output yang ingin lulus.*
`int totalStemmedWords;`	`4`
`int totalWordsIn;`	`18`
`int totalOutput;`	`18`
Tagging Testing	Worked properly
Case sensitive testing	Worked properly

Table 6.1 : Tested result

6.3 System & Integration Testing

The system had been tested for the unit testing. All input will produce the correct output after modification. Integration testing is more to the system for this project. This is because the main purpose is going to be used by the other peoples to build their own systems. If they got no problem of reusing this component, meaning that this component can be integrated with other applications. Salhana bt. Darwin had been build a system "Essay Grading Using Nearest Neighbor Technique" and implemented this component-based. She got no problem to use this component integrated with her project.

6.4 Conclusion

Resulting from all the tests that had been carried out, it seem that all modules can perform well but still have several problems to make it really useable for all language. These conditions arise due to the several problems, which will be discussed in the next chapter. However, all those modules are not 100 percent fail. If all the problems that caused those modules unable to fulfill their functional requirement could be overcome, it believed that all modules could perform their functional requirement very well.

CHAPTER 7

SYSTEM DISCUSSION

7.0 System Discussion

7.1 Introduction

In this chapter, it will be a discussion on the usage of stemming component and the conclusion of the test result. In this chapter also, will be discussed a knowledge that has been found from a research on the ability of this component and what is the future enhancement that can be implemented.

7.2 Discussion On The Module Test Result

The test result on all modules seems returned a recommended output. But it still faced with a problem where the external file (data file) have to sort properly and manually with the longest length of words/prefix/suffix/infix at the top of the file.

7.3 Handling Visual C++

As a new tool for programming, it took quite some time for me to learn the feature in visual C++ and what it is capable of doing. The main problem that I faced was to how to use Visual C++. As resources for learning Visual C++ were not enough, even though I had several books on Visual C++, it is still not enough, this made the process of learning even harder. Because of that, some objectives were failed to develop such as integrate with fancy user interface and integrate with other programming languages.

To integrate with other programming languages such as Visual Basic, Visual J++ and ASP, some additional new language have to learn which is known

ATL(*Active Template Library*). This ATL use C++ as the core language. With ATL we can compile and register dll(*dynamic link library*) files to the system and easy to call the component. It can reduce response time and give a good performance. How ever it is not as easy to learn ATL. This is because normally advance programmers used this ATL to combine or integrate their system with difference programming language. So, the developers should know a lot of language. It is so difficult for me to learn everything about ATL to build advance system such as this component-based where it can be used for many development platforms.

7.4 System Strengths

7.4.1 Reusable of code

The main objective of component-based is can be used by other application. It seems like object-oriented approach. This component can be plugged into any system or attached into other system programs. It will work properly without any changes to the component.

7.4.2 Fast Response Time for Document retrieval

When the stem function is invoked, relatively the time taken to process the word and displaying the results is quite fast. Even be it a word, sentences or text file with full text, the response time is still fast.

7.4.3 Multi Language Approach

The main purpose of this component is for Malay text usage. But it can be implemented to any language or new language since it is text format. The end users only have to declare their own language set of dictionary, prefixes, suffixes, infixes and the exceptional cases.

7.5 System Constraints

7.5.1 Visual C++

For this purpose, this component only runs in computers that are pre installed with Visual C++ 6.0. This is because the unavailability of the creating setup files for Visual C++.

7.5.2 Exceptional Cases

There are several exceptional cases in the Malay language words like for example a word can be stemmed into have root words, which both of them are correct. This case is very clear for the other language such as English, which there are so many exceptions in their morphology.

7.6 Future Enhancement

Since the system that being developed is still just a prototype, it is future plan to make this component-based stemming engine as a real time application with full friendly user interface. Mean that the users will be the Malay Language Researchers or other language researchers and the system will be used fully for research enhancement of the language.

7.6.1 Essay Answer Marking

In essay grading system, stemming is very useful to implement. It is important because the student's answer will have a lot of non-root words where the words should be matched with the teacher's schema.

7.6.2 Language Based Intelligent System

This component can be used into intelligent system such as a document filtering for file or email transaction. Searching engine also using stemmed word to retrieve matches keyword currently. This component-based stemming engine approach also could be upgraded to use for spelling checker agent like Microsoft Word to check and correct words or sentences in the document file.

7.6.3 Implementing Database

Converting the existing storage format from file storage, to a database storage for better performance since if the system is further being enhanced the system should be able to cater larger data of the dictionary, and if we are using file storage, it will be set back as it will degrade the performance of the system during enhancement process.

CONCLUSION

Conclusion

This system is developed to serve a new Malay words stemming system's application especially for a specific domain. It caters for all types of Malay modern words only. However the system will be used as a prototype for the purpose of experiments for further enhancements of the Malay words stemmer component, in order to reduce the errors occurred.

The component-based technology that has been applied to this stemming engine cause many others applications can be build easily, just simply put certain source code of this component to their own applications. It is also can be used for manipulating or developing other language since it is a text format.

This system has its advantages and weakness but it still satisfied the specification of the system. This system is developed to provide help for students who are doing research in stemming process especially in Malay Language, and other people who are involved or in need of this stemming process service.

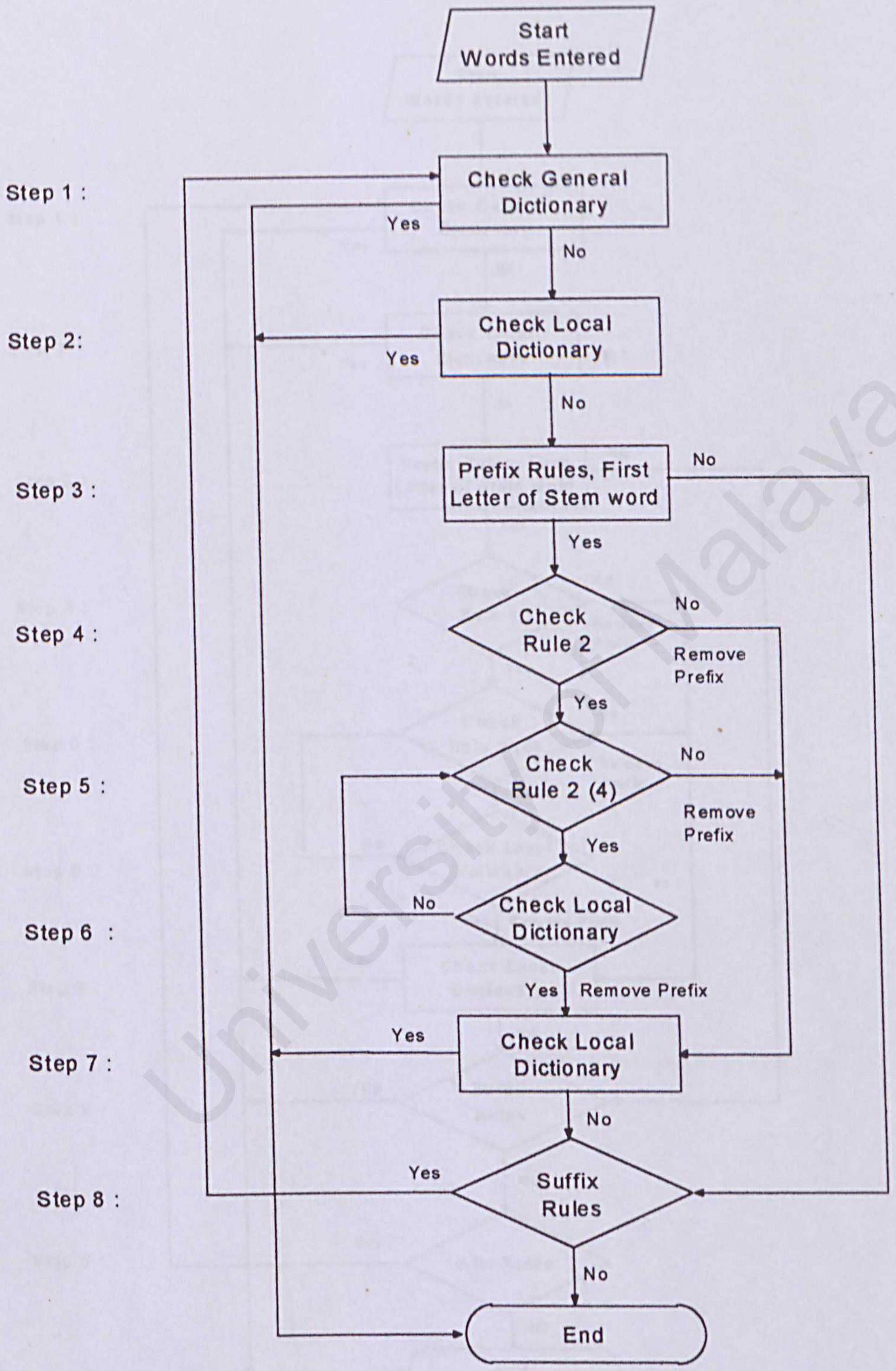
A lot of knowledge was gained throughout the development of the system. These include knowledge of programming in Visual C++, using Visio Professional, and ATL.

Finally, all the problems faced and experiences gained during the system development should be useful for future endeavors.

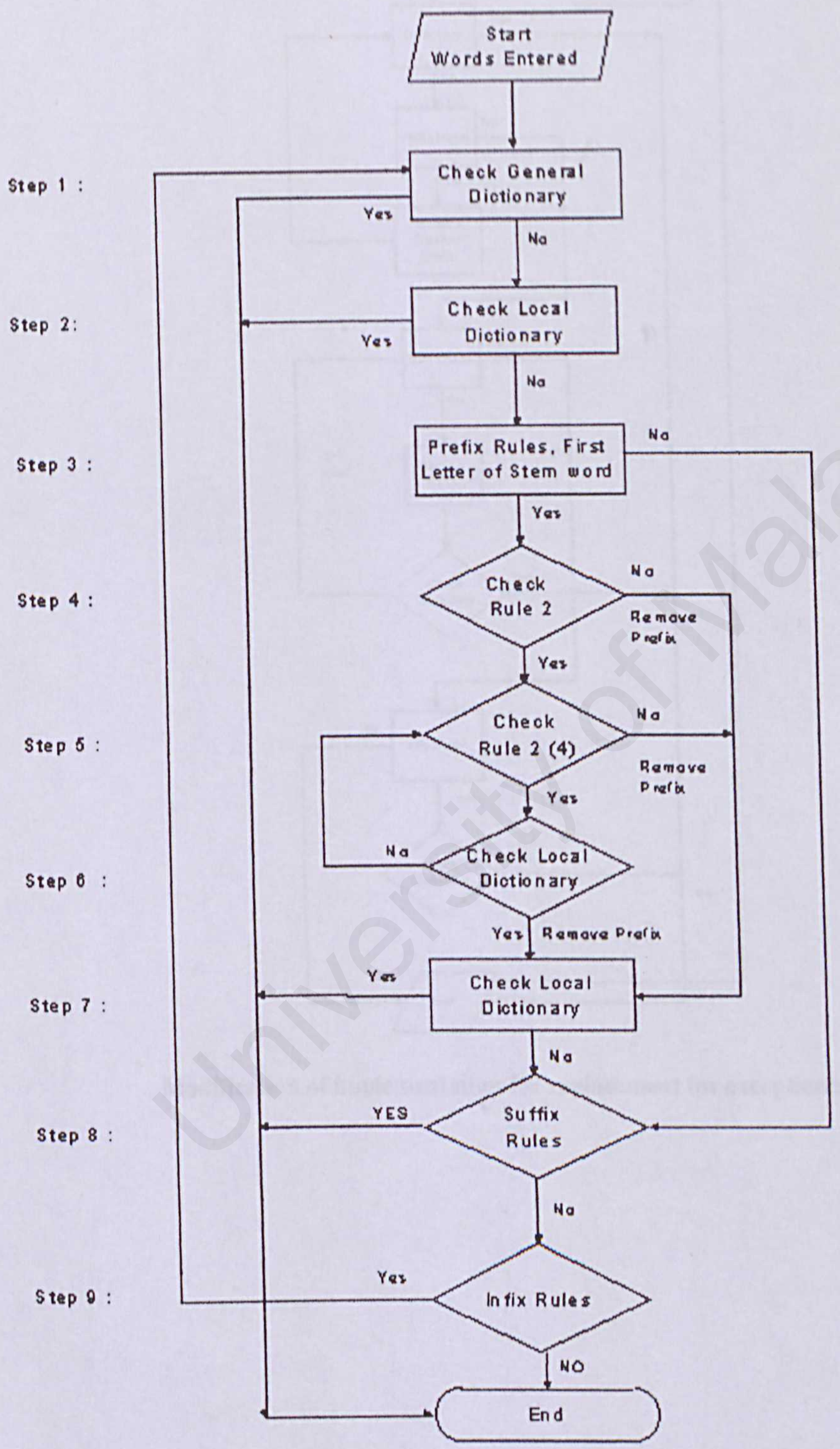
APPENDIX

APPENDIX A

Flow Chart of the Malay Stemming Alogrithm

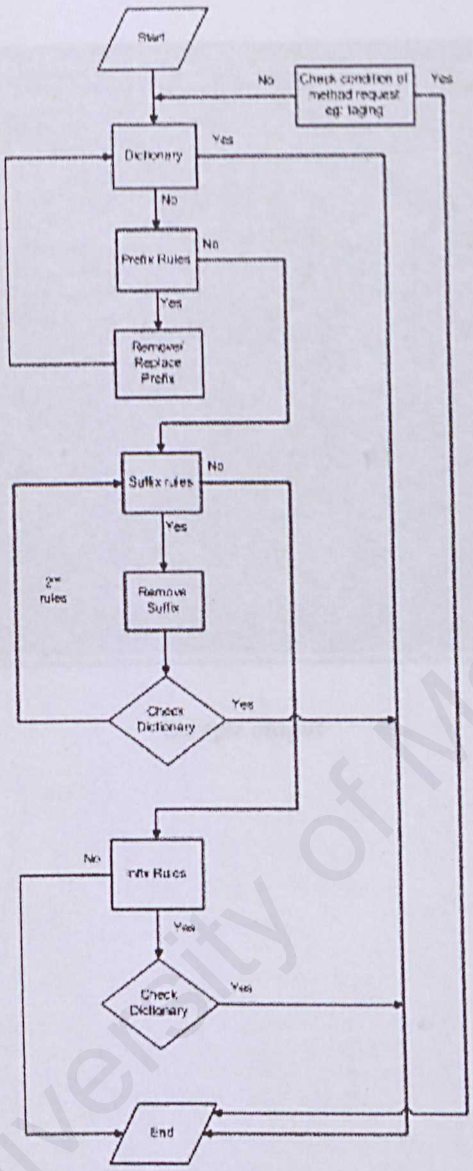


APPENDIX B



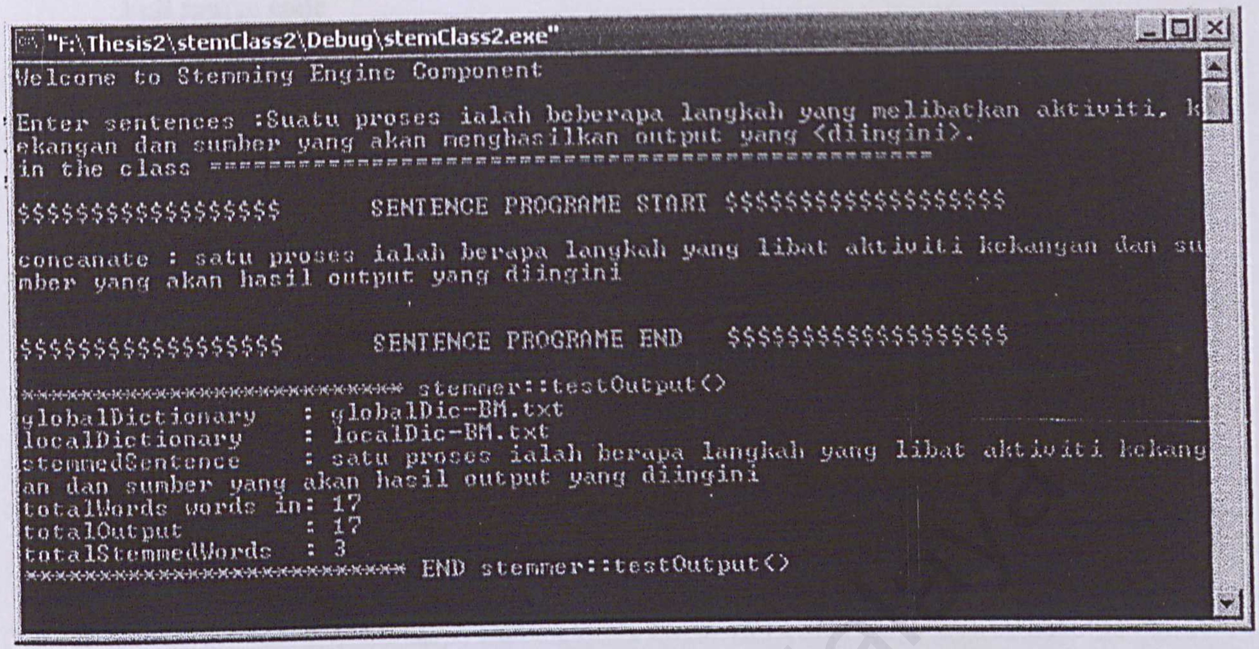
Modification from Somabalan[1]

APPENDIX C



Modification of implementation for replacement for exceptional cases

APPENDIX E



```
"F:\Thesis2\stemClass2\Debug\stemClass2.exe"
Welcome to Stemming Engine Component

Enter sentences :Suatu proses ialah beberapa langkah yang melibatkan aktiviti, k
ekangan dan sumber yang akan menghasilkan output yang <diingini>.
in the class ================================================

$$$$$$$$$$$$$$$$$$      SENTENCE PROGRAME START $$$$$$$$$$$$$$$$$$$$

concanate : satu proses ialah berapa langkah yang libat aktiviti kekangan dan su
mber yang akan hasil output yang diingini


$$$$$$$$$$$$$$$$$$      SENTENCE PROGRAME END   $$$$$$$$$$$$$$$$$$$$

*************************** stemmer::testOutput()
globalDictionary   : globalDic-BM.txt
localDictionary    : localDic-BM.txt
stemmedSentence    : satu proses ialah berapa langkah yang libat aktiviti kekang
an dan sumber yang akan hasil output yang diingini
totalWords words in: 17
totalOutput        : 17
totalStemmedWords  : 3
*************************** END stemmer::testOutput()
```

Sample output

APPENDIX F

Full source code

File Name : stemmerMain.cpp

```
#include<iostream.h>
#include<string.h>
#include<fstream.h>
#include<afx.h>
#include<stdio.h>
#include<conio.h>

#include "stemmClass.h"


void main()
{

	cout<<"Welcome to Stemming Engine Component "<<endl<<endl;
	fflush(0);

			const int sentenceLen =300;
			char sentenceInput[sentenceLen];	//sentences entered
			CString sentenceOut;			// stemmed
sentences
			stemmer obj1;

			cout<<"Enter sentences :";
			cin.get(sentenceInput,sentenceLen,'\n');

			char word4[sentenceLen];
			int panjang4=strlen(sentenceInput);
			strncpy(word4,sentenceInput,panjang4);
			word4[panjang4]='\0';

			CString sentenceInput2 = word4;
			sentenceOut ="dummyX " + sentenceInput2;

			cout<<"in the class
==================================================="<<endl;

			char word3[sentenceLen];
			int panjang=strlen(sentenceOut);
			strncpy(word3,sentenceOut,panjang);
			word3[panjang]='\0';

			obj1.stem(word3,0,0);
			/*
			Will have a default

			1st arg : Sentence input
			2st arg : 0/not sensitive
			          1 case sensitive
			3st arg : 0 Default : stem symbol n not stem tagging
			          1 stem symbol only
			          2 stem tagging and symbol
			*/
			obj1.testOutput();
			cout<<endl;

		cout<<endl;

}


stemmer::stemmer(CString global, CString local,CString preffile,CString
suffile,CString inffile)
{
	fileGlobalDictionary = global;
	fileLocalDictionary = local;
	filePrefixes = preffile;
	fileSuffixes = suffile;
	fileInfixes  = inffile;
	totalStemmedWords=0;
	totalWordsIn=0;
	totalOutput=0;
	stemmedSentence = "";
	wordIn="";
	rootword="";


}

void stemmer::testOutput()
{
	cout<<"************************** stemmer::testOutput()"<<endl;
	cout<<"globalDictionary	: " <<fileGlobalDictionary<<endl;
	cout<<"localDictionary	: " <<fileLocalDictionary<<endl;
	cout<<"stemmedSentence	: " <<stemmedSentence<<endl;

	cout<<"totalWords words in: " <<totalWordsIn<<endl;
	cout<<"totalOutput		: " <<totalOutput<<endl;
	cout<<"totalStemmedWords	: " <<totalStemmedWords<<endl;

	cout<<"************************** END stemmer::testOutput()"<<endl;
}

void stemmer::stem(char *sentence,int caseSensetive,int tag)
{
	cout<<endl<<"$$$$$$$$$$$$$$$$$$	SENTENCE PROGRAME START
	$$$$$$$$$$$$$$$$$$"<<endl<<endl;

	static CString concanate ;
	char *token;

	CString sym;
	if (tag==0)
		sym = " ,.\t\n;':*$@";
	else if (tag==1)
		sym = " ,.\t\n;':*$@";
	else if (tag==2)
		sym = " ,.\t\n;':*$@<>";
	else
		sym = " ,.\t\n;':*$@";


    /* Establish string and get the first token: */

    token = strtok( sentence, sym );

	 static int totalWords=0;
	 static int totalOutputWords=0;

   while( token != NULL )
   {
      /* While there are tokens in "string" */
      /* Get next token: */
      token = strtok( NULL, sym );
	  CString tokWord=stemWord(token,caseSensetive,tag);
	  if (tokWord != "")
	  {
		  concanate = concanate + tokWord + " ";
		  totalOutputWords = totalOutputWords + 1 ;

	  }
	  totalWords = totalWords + 1 ;
	  totalOutput = totalOutputWords;
    }

	stemmedSentence = concanate;
	totalWordsIn = totalWords - 1;


   cout<<"concanate : "<<concanate<<endl<<endl;
```

```
        cout<<endl<<"$$$$$$$$$$$$$$$$$$        SENTENCE PROGRAME END
        $$$$$$$$$$$$$$$$$$$"<<endl<<endl;
}

CString stemmer::stemWord(CString word,int caseSensetive,int tag)
{
        int inGlobal;

        CString rootWord;
        CString rootLocalTemp;
        if (caseSensetive == 0)
                word.MakeLower();


        CString leftChr = word.Left(1);
        CString RightChr = word.Right(1);
        if (leftChr=="<" && RightChr ==">")
        {
                word.Remove('<');
                word.Remove('>');
                rootWord = word;
                return rootWord;
        }

        inGlobal = globalDictionary(word);    // will return TRUE if exist
        if (inGlobal==TRUE)
        {
                rootWord = word;
        }
        else
        {       //if not found in the global then try to find in the local
                rootLocalTemp = localDictionary(word); // will return replacement of
rootword if found otherwise null
                rootWord=rootLocalTemp;

        }
        /*      if rootWord="" then go to prffix
                prefix will return the stemmed word

        */
        CString wordPref="";
        CString wordSuffix="";
        CString wordInfix="";


        if(rootWord=="") // GOTO PREFIX
        {
                //int stemmedPrefIndex;

                wordPref = Prefix(word);
                int inGlobalPref = globalDictionary(wordPref);          // will return
TRUE if exist
                if (inGlobalPref==TRUE) {
                        rootWord = wordPref;
                        totalStemmedWords = totalStemmedWords + 1;
                }
                else
                {       //if not found in the global then try to find in the local
                        CString prefLocalTemp = localDictionary(wordPref); // will
return replacement of rootword if found otherwise null
                        rootWord=prefLocalTemp;
                }
        }
        if(rootWord=="") // GOTO SUFFIX
        {
                if (wordPref=="")
                        wordSuffix = Suffix(word);
                else
                        wordSuffix = Suffix(wordPref);

                if (wordSuffix!="")
                {
                        int inGlobalSuff = globalDictionary(wordSuffix);     // will
return TRUE if exist
                        if (inGlobalSuff==TRUE) {
```

```
                                        rootWord = wordSuffix;
                                        totalStemmedWords = totalStemmedWords + 1;
                                }
                                else
                                {       //if not found in the global then try to find in the
local
                                        CString suffLocalTemp = localDictionary(wordSuffix); //
will return replacement of rootword if found otherwise null
                                        rootWord=suffLocalTemp;
                                }
                        }
                }
                if (rootWord=="")
                {
                        CString wordOri = word;
                        CString wordStem1 = wordPref;
                        CString wordStem2 = wordSuffix;

                        if (wordStem1=="" && wordStem2=="")
                                wordInfix = Infix(word);
                        else if (wordStem1!="" && wordStem2=="")
                                        wordInfix = Infix(wordStem1);
                        else if (wordStem1=="" && wordStem2!="")
                                        wordInfix = Infix(wordStem2);
                        else if (wordStem1!="" && wordStem2!="")
                                        wordInfix = Infix(wordStem2);

                        if (wordInfix!="")
                        {
                                int inGlobalInf = globalDictionary(wordInfix);          // will
return TRUE if exist
                                if (inGlobalInf==TRUE) {
                                        rootWord = wordInfix;
                                        totalStemmedWords = totalStemmedWords + 1;
                                }
                                else
                                {       //if not found in the global then try to find in the
local
                                        CString suffLocalTemp = localDictionary(wordInfix); //
will return replacement of rootword if found otherwise null
                                        rootWord=suffLocalTemp;
                                }
                        }
                }
                return rootWord;
}

int stemmer::globalDictionary(CString word)
{
        int found = 0;
        int counter=0;
        char globalRoot[30];

        CString rootTemp;
        CString wordTemp;

        ifstream GlobalFile(fileGlobalDictionary,ios::in);                              // will be
dynamically
        if (!GlobalFile)
                cout<<"File cannot be opened..."<<endl;
        else
                while ( found!=1 && GlobalFile.peek()!= EOF)
                {
                        GlobalFile >> globalRoot;
                        counter=strcmp(word,globalRoot);
                        if(counter == 0)
                        {
                                found=1;
                                rootTemp = globalRoot;
                        }
                        else
                                found=0;
                }
        return found;
```

```
}

CString stemmer::localDictionary(CString word)
{
	int found=0;
	int counter=0;
	char localRoot[30],localRootReplace[30];
	CString wordTemp=word;
	CString rootTemp;

	ifstream LocalFile(fileLocalDictionary,ios::in);	// will be dynamically
	if(!LocalFile)
		cout<<"File local not found";
	else
		while (found!=1 && LocalFile.peek()!=EOF )
		{

			LocalFile>>localRoot>>localRootReplace;
			counter=strcmp(wordTemp,localRoot);
			if(counter==0) // 0 : the string is match
			{
				rootTemp = localRootReplace;	// replace the word
				found=1;

			}
			else
				found=0;
		}
	return rootTemp;
}

CString stemmer::Prefix(CString wd)
{
	CString word = wd;
//	cout<<"*************************************** IN the PRefix()"<<endl;
	int lenPrefixes;
	int found=0;

	char prefixesTemp[10];			// preffix from txt file
	char prefixesReplaceTemp[10]; // replacement of preffix from txt file


	CString prefixes,prefWord,rootTemp;

	ifstream PrefFile(filePrefixes,ios::in);
	if (!PrefFile)
		cout<<"File cannot be opened..."<<endl;
	else
	{
		while ( found!=1 && PrefFile.peek()!=EOF)
		{
			PrefFile>>prefixesTemp>>prefixesReplaceTemp;
			prefixes=prefixesTemp;

			lenPrefixes = prefixes.GetLength();		// try to get
 Prefix len from prefix file
			prefWord = word.Left(lenPrefixes);		// try to get
 Prefix for word
			if (strcmp(prefixes,prefWord)==0)
				found = 1;
	//prefix found=1
			else
				found = 0;
		}
	}
		if (found == 1)
		{

			rootTemp=word.Mid(lenPrefixes);
			CString Vowel = rootTemp.Left(1);	// try to get a vowel

			if ((Vowel=="a" )  || (Vowel=="e") || (Vowel=="i") ||
 (Vowel=="o") || (Vowel=="u"))
			{
				if(prefixesReplaceTemp!="**")
```

```
                                        rootTemp=prefixesReplaceTemp + rootTemp;
                        }

                }
        return rootTemp;
//      cout<<"****************************************** END of the PRefix()"<<endl;
}

CString stemmer::Suffix(CString word)
{
//      cout<<"****************************************** START SUFFIX()"<<endl;
        CString wordTemp;
        wordTemp = word;

        int lenSuffixes;
        int found=0;

        char SuffixesTemp[10];                  // preffix from txt file
//      char SuffixesReplaceTemp[10]; // replacement of preffix from txt file

        CString Suffixes,suffWord,rootTemp;

        ifstream SuffFile(fileSuffixes,ios::in);
        if (!SuffFile)
                cout<<"File cannot be opened..."<<endl;
        else
        {
                while ( found!=1 && SuffFile.peek()!=EOF)
                {
                        SuffFile>>SuffixesTemp;//>>SuffixesReplaceTemp;
                        Suffixes=SuffixesTemp;

                        lenSuffixes = Suffixes.GetLength();             // try to get
Suffixes len from prefix file
                        suffWord = wordTemp.Right(lenSuffixes);                 // try to
get Suffixes for word
                        //cout<<"suffix:"<<Suffixes<<endl;
                        if (strcmp(Suffixes,suffWord)==0)
                                found = 1;
        //Suffixes found=1
                        else
                                found = 0;
                }
        }
                if (found == 1)
                {
                        int lenWordTemp = wordTemp.GetLength();
                        int lenRootWord= lenWordTemp - lenSuffixes;

                        rootTemp=wordTemp.Left(lenRootWord);
                }
//      cout<<"****************************************** END SUFFIX()"<<endl;

        return rootTemp;
}

CString stemmer::Infix(CString word)
{
//      cout<<"****************************************** START INFIX()"<<endl;
        CString wordTemp;
        wordTemp = word;

        int lenInfix=0;
        int found=0;

        char InfixTemp[10];                     // preffix from txt file
//      char SuffixesReplaceTemp[10]; // replacement of preffix from txt file

        CString Infixes, rootTemp;


        ifstream InfFile(fileInfixes,ios::in);
        if (!InfFile)
                cout<<"File cannot be opened..."<<endl;
        else
        {
              while ( found!=1 && InfFile.peek()!=EOF)
              {
                      InfFile>>InfixTemp;//>>InfixesReplaceTemp;
                      Infixes=InfixTemp;
                      if (wordTemp.Find(Infixes) >= 0)
                      {
                              lenInfix=Infixes.GetLength();
                              found = 1;
      //Infixes found=1
                      }
                      else
                              found = 0;
              }
      }
              if (found == 1)
              {
                      int posInfix=wordTemp.FindOneOf(Infixes);
                      int lenWord = wordTemp.GetLength();
                      CString leftWord = wordTemp.Left(posInfix);

                      int lenLeftWord=leftWord.GetLength();
                      CString rightWord = wordTemp.Right(lenWord-lenInfix-
lenLeftWord);
                      rootTemp = leftWord + rightWord;
              }
//      cout<<"***************************************** END INFIX()"<<endl;
        return rootTemp;
}
```

File Name : stemmClass.h

```
class stemmer
{
private:
        CString fileGlobalDictionary, fileLocalDictionary;
        CString filePrefixes, fileSuffixes , fileInfixes;
        CString wordIn, rootword;
        CString sentenceIn;
        CString stemmedSentence;
        int totalStemmedWords;
        int totalWordsIn;
        int totalOutput;

public:
        stemmer(CString ="globalDic-BM.txt", CString ="localDic-BM.txt",
CString="prefixes-BM.txt", CString="suffixes-BM.txt", CString="infixes-BM.txt");
        void testOutput();
        void stem(char *sentence,int caseSensetive=0,int tag=0);
        CString stemWord(CString word,int caseSensetive=0,int tag=0);
        int globalDictionary(CString word);
        CString localDictionary(CString word);
        CString Prefix(CString word);
        CString Suffix(CString word);
        CString Infix(CString word);

};
```

APPENDIX F

Component Based Stemming Engine for Malay Text

Overview of the 1st proposal (WXES3181)

1. Introduction of Stemming Engine
    a. Process of extracting each word from text document, reducing it to a probable root word.
    b. Technique of linguistic normalization, in which the variant forms of a word are reduced to a common form.
    c. Removing affixes from the text document or query produces a stemmed word.
    f. Affix is the verbal elements that attached to the beginning of the word(*prefix*), end of the word(*suffix*) and in the middle of the word(*infix*).
2. Objectives
    a. To provide a global component which is reusable for developers to build their own application/system.
    b. Can be used for all MS Window application.
    c. To develop IR for Malay text and the morphology.
    d. To be used in education field
        i. Writing exam online
        ii. Automated essay grading
3. Component Implementation
    a. Component system is designed to facilitate reuse of code. Component-based development assumes that the developer will create an application by writing together components that have already been created by someone else.
    b. Module of code that is created with the intention that other developer will plug them into new applications.
    c. The code always stays static
        i. A component is developed and the code there after remain unchanged.
        ii. An object can be considered as an instantiation of a component.
4. Functionality
    a. OOP Based – Can be used to every developer.
    b. Advance searching – Searching engine for web-based / dictionary.
    c. Education – Essay grading.
    d. Can be used to many platform application;
        i. Web Based (ASP, JSP, PHP).
        ii. System (Visual C++, Visual Basic)
        Operating System (MS Windows, Unix/Linux) Why ? * Using Binary code after compiled to specific file (.dll,exe).

Presentation 2

1. Introduction of Component based Stemming Engine for Malay text.
2. System flow diagram.
3. System achievement.
    - Generally this system is successful.
        - Reusable component.
        - Multilanguage processing.

4. Stemming Component

   Class Name :

   stemmer(CString ="globalDic-BM.txt", CString ="localDic-BM.txt")
   stem(char *sentence,int caseSensetive=0,int tag=0)
   stemWord(CString word,int caseSensetive=0,int tag=0);
   testOutput()
   stemmer.totalStemmedWords;
   stemmer.totalWordsIn;
   stemmer.totalOutput;

```
class stemmer{
private:
    CString fileGlobalDictionary, fileLocalDictionary;
    CString filePrefixes, fileSuffixes , fileInfixes;
    CString wordIn, rootword;
    CString sentenceIn;
    CString stemmedSentence;
    int totalStemmedWords;
    int totalWordsIn;
    int totalOutput;
public:
    stemmer(CString ="globalDic-BM.txt", CString ="localDic-BM.txt");
    void testOutput();
    void stem(char *sentence,int caseSensetive=0,int tag=0);
    CString stemWord(CString word,int caseSensetive=0,int tag=0);
    int globalDictionary(CString word);
    CString localDictionary(CString word);
    CString Prefix(CString word);
    CString Suffix(CString word);
    CString Infix(CString word);
};
```

5. Constrain
    a. Multilanguage processing – Problem with the special word to stem for their morphology.
       eg: BM : menyedari → sedar.
       BI : too many exception.
       broke → break, ran → run, etc.
    b. Multilanguage programming – ATL(*Active Template Library)* Need a professional/expert programmer.
6. Future enhancement & opinion

a. Multilanguage programming
b. ATL

References

[1] Somabalan, R.,(2001), Word Stemmer Application, Degree Thesis. University of Malaya.

[2] Norisma Idris, Automated Essay Grading System Using Nearest Neighbor Technique, Degree Thesis. University of Malaya.

[3] Buku Software Engineering

[4] Kozaczynski, W., and Booch, G.: 'Component-based software engineering', IEEE Software, October 1988, pp.34-36

[5] Krutchen, P.: Rational Software, http://www.rational.com/sitewide/support/resources.jtmpl (July 2002)

[6] Szperski, C.: 'Emerging component software technologies – a stratergic comparison', Software-Concept & tools, 1998, **19**, (1), pp. 2-10

[7] Harmon, P. : Cutter Information Corp, http://www.cotter.com (July 2002)

[8] Loureiro, K.,and Blechar, M.: 'Componentware: Categorization cataloguing', Applications Development and Management Strategies Research Note, Gartner Group, December 1997

[9] Riverton Software Corp. http://www.riverton.com/solutions/knowhow/tlassets.htm (July 2002)

[10] Chappel, D.: http://www.chappellassoc.com/article.htm (July 2002)

[11] Plassil, F., and Stal, M.: 'An Architectural view of distributed objects and and component in COBRA, Java RMI and COM/DCOM', *Software – Concepts and Tools*, 1998, **19**, (1), pp.14-28

[12] Hull, M.E.C., Nicholl, P.N. , : 'Approaches to component technologies for software reuse of legacy system', *Computing & Engineering Journal*, 2002, 12,(2), pp.281-287

[13] Dewan Bahasa dan Pustaka, Tatabahasa Dewan, Edisi Kedua, Kementerian Pendidikan Malaysia, Kuala Lumpur, 1999.

[16] M.F. Porter, An Algorithm for Suffix Stripping Program, 1980, Volume 14, Number 3, Pages 130-137.

[17] Visual C++ example codes
http://www.codeguru.com (Dec 2002)

[18] Microsoft Visual C++
http://www.support.microsoft.com/visualc/ (Dec 2002)

USER MANUAL

Introduction

This component-based stemming engine is created for default language as Malay. Its main objective's are to improve the efficiency of the Information Retrieval of a full texts Malay database. It is hopefully to be used in a lot of application areas such as essay grading for school usage, advance searching and intelligent filtering agent. This is just a simple application to view the output of the component. However, it can be used to upgrade and implement into user interface module.

Application Usage Guideline

This component-based is very useful for developers. It is not really easy to use for the end user who does not have experience with Visual C++. This component code has to add several functions to make it more user friendly.

Basic Requirements

Hardware requirement;

1. Computer 'IBM Compatible' with at least Pentium Processor with 133Mhz.
2. 32 MB RAM or over.
3. 500Mb hard disk space for installing important software not included the Operatimng System.
4. Mouse.
5. CD-ROM
6. VGA Monitor.

Software requirement;

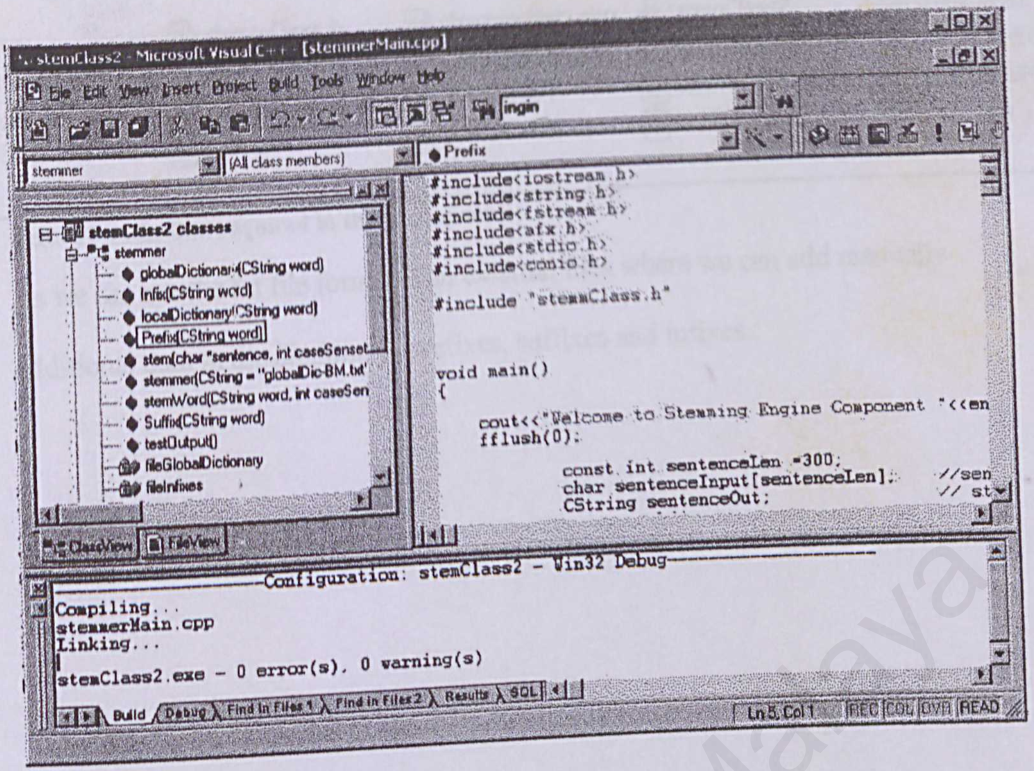
1. Win95 OS or latest Microsoft Windows OS.
2. Microsoft Visual C++ 6.0

Executing component-based stemming engine

1. Copy **stemClass** folder from CD to any folder into your hard disk.
2. Open Visual C++ application.
3. Open the workspace

   file→open workspace→ stemClass2.dsw (in the stemClass folder from your hard disk).
4. Compile and running the application.

As you can see the object **obj1** is created in the main function as **stemmer** and using default value for the constructor's attributes. It is meaning that Malay Language is being used for the application. **obj1.stem(word3,0,0)** is an object to send a words or sentences to stem function to get the stemmed output. **obj1.stemmedSentence** will return stemmed sentences with default option.

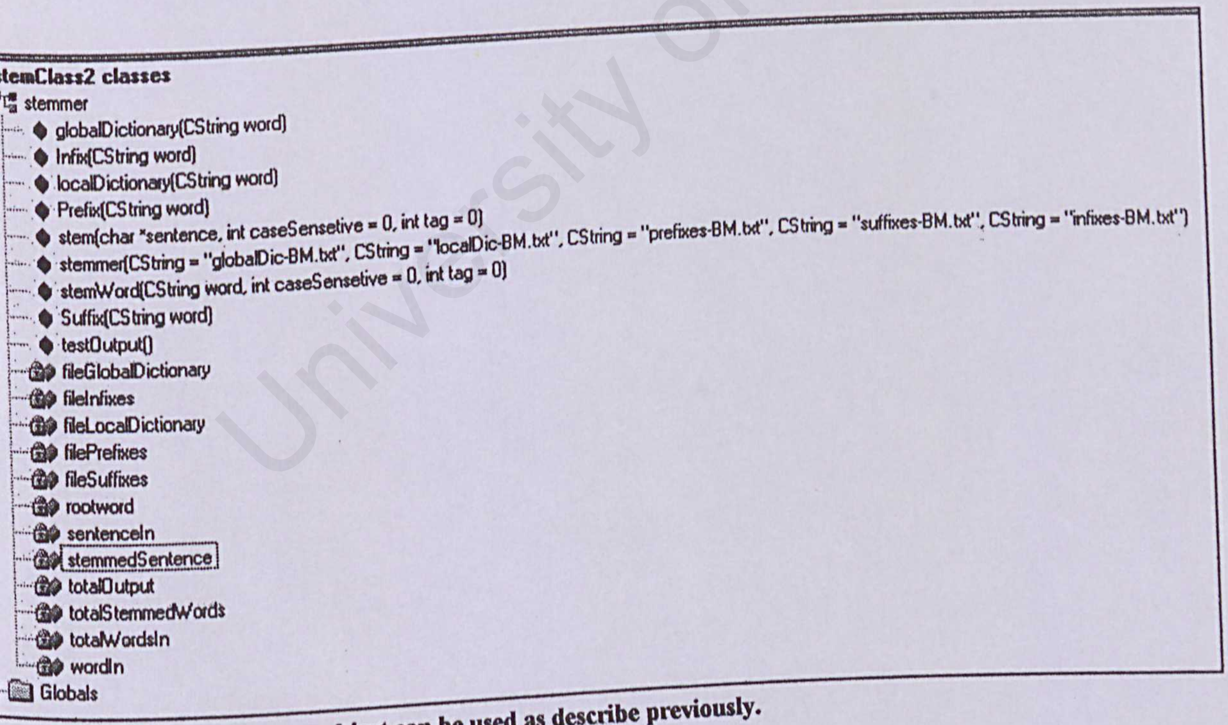


```
stemClass2 classes
  stemmer
    globalDictionary(CString word)
    Infix(CString word)
    localDictionary(CString word)
    Prefix(CString word)
    stem(char *sentence, int caseSenset
    stemmer(CString = "globalDic-BM.txt'
    stemWord(CString word, int caseSen
    Suffix(CString word)
    testOutput()
    fileGlobalDictionary
    fileInfixes
```

```
#include<iostream.h>
#include<string.h>
#include<fstream.h>
#include<afx.h>
#include<stdio.h>
#include<conio.h>

#include "stemmClass.h"


void main()
{

    cout<<"Welcome to Stemming Engine Component "<<en
    fflush(0);

          const int sentenceLen =300;
          char sentenceInput[sentenceLen];      //sen
          CString sentenceOut;                  // st
```

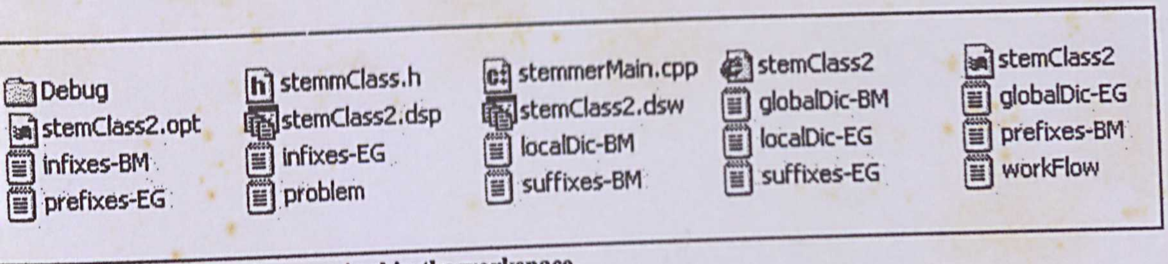
```
--------------------Configuration: stemClass2 - Win32 Debug--------------------
Compiling...
stemmerMain.cpp
Linking...

stemClass2.exe - 0 error(s), 0 warning(s)
```

Figure 1 shows workspace with visual C++



```
temClass2 classes
  stemmer
    globalDictionary(CString word)
    Infix(CString word)
    localDictionary(CString word)
    Prefix(CString word)
    stem(char *sentence, int caseSensetive = 0, int tag = 0)
    stemmer(CString = "globalDic-BM.txt", CString = "localDic-BM.txt", CString = "prefixes-BM.txt", CString = "suffixes-BM.txt", CString = "infixes-BM.txt")
    stemWord(CString word, int caseSensetive = 0, int tag = 0)
    Suffix(CString word)
    testOutput()
    fileGlobalDictionary
    fileInfixes
    fileLocalDictionary
    filePrefixes
    fileSuffixes
    rootword
    sentenceIn
    stemmedSentence
    totalOutput
    totalStemmedWords
    totalWordsIn
    wordIn
Globals
```

Figure 2 shows an abject can be used as describe previously.



Debug
stemClass2.opt
infixes-BM
prefixes-EG
stemmClass.h
stemClass2.dsp
infixes-EG
problem
stemmerMain.cpp
stemClass2.dsw
localDic-BM
suffixes-BM
stemClass2
globalDic-BM
localDic-EG
suffixes-EG
stemClass2
globalDic-EG
prefixes-BM
workFlow

Figure 3 : File that required in the workspace.

As we can see the txt file format is an external files where we can add manually additional data in the dictionary, prefixes, suffixes and infixes.