Asian Journal of Information Technology 5 (3) : 272-279, 2006

A Tool for Translating A Software Specification Written in Malay into A Formal Statements in Z

Zarina Shukur, Abdullah Md. Zin, Ainita Ban and See Cze Ping
Programming Research Group, Fakulti Teknologi dan Sains Maklumat,
Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor

Abstract: This study discusses the design and implementation of a tool for translating a natural language software specification into a formal specification. The input to the tool are basic information about the system to be specified and a statement describing the specification of the system written in the Malay language. This language is widely spoken in South East Asia. The basic information is used as the basic knowledge about the system. By using this basic knowledge, a specification statement will be translated and semantically interpreted in order to produce an equivalent statement in Z.

Key words: Formal methods, machine translation, linguistic, Z

INTRODUCTION

Software specification is normally written in a natural language. As with other documents that are written in a natural language, a software specification normally has a lot of ambiguouty, especially when it is read and interpreted by different people. The misunderstanding of a software specification due to these ambiguouties has been identified as one of the most important source of error in software development. For example, Boehm has stated that more than 60% of errors in software development is due to the error in understanding the software specification[1].

In order to reduce the possibilities of errors in a software specification a number of proposals have been put forward. One of the proposals is to write software specifications by using programming language. However it was found that programming language is not suitable for writing a software specification since programming language focuses on how to solve the problem and not on what to to be solved[2].

Formal notations that are based on mathematics, such as Z and VDM, have been considered to be more effective in representing software specifications. Although the benefit of using formal notations is generally accepted by most of software practioners, formal notations are not widely used in software development[3]. Most of software developers are not familiar with mathematical notation and they find that writing matematical statements is too complicated[4].

A possible solution to this problem is by providing software developers with a tool that can aid in translating natural language statements into mathematical statements.

A few such tools have been developed over the last decade, for examples SPECIFIER[5], RA[6], NL2ACTL[7] and FORSEN[8]. A more recent effort in this area involves translation of requirement documentation to an object-oriented formal specification language[9]. Another research involves developing a tool for linking formal specification of programs written in OCL to informal specifications written in natural language such as English or German[10].

The decription of some of these tools is shown in Table 1. From Table 1, we can see that all of these tools were designed to translate natural language statements in English into statements in formal notation.

This study discusses the design and implementation of a tool that can help in translating translate a natural language statement written in the Malay language into a formal specification statement in Z. The Malay language is spoken by more than 200 million people in South East Asia. This paper is divided into five sections. The second section of the paper explains the concept of Machine Translation (MT). The third section describes some of the systems for translating natural language statements that have been developed. The fourth section deals with the design of M2Z while the last section is the conclusions.

MACHINE TRANSLATION (MT)

Machine Translation (MT) is a computer-based process that translates statements from one language into

Corresponding Author: Zarina Shukur, Programming Research Group, Fakulti Teknologi dan Sains Maklumat, Universiti Kebangsaan Malaysia, 43600 Bangi, Selangor

Table 1: Tools for formalising natural language

Tools	Researchers	Source language	Target language
Specifier	Miriyala and Harandi	English	LarchSharedLanguage
RA	Reubenstein and Waters	English	More formal language
NL2ACTL	Paolo Moreschine *et al.*	English	Temporal Logic formulae
FORSEN	Vadera and Meziane	English	VDM

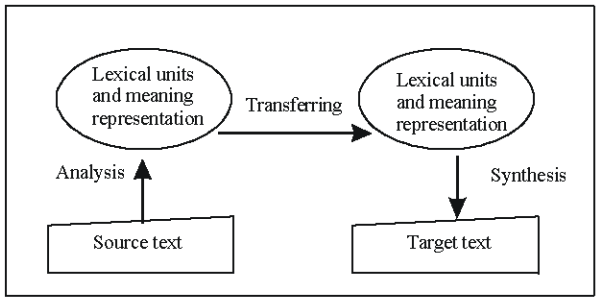

Fig. 1: Transfer approach

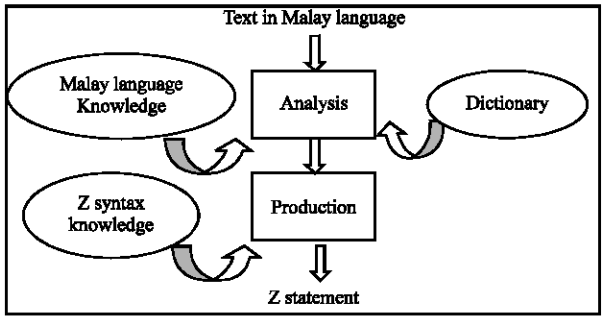

Fig. 2: Architecture of M2Z

another language[11]. The initial idea of MT started in the middle of 1950s. Initially this concept suffered a major setback. In 1959, a philosopher by the name of Bar Hillel, stated that Machine Translation (MT) is impossible because computer is unable to think[11,12]. However, the advancement of research in linguistics, artificial intelligence and technological tools has enabled a number of MT systems to be developed.

MT architecture can be divided into two types of architecture, which are direct and indirect. In direct translation architecture, input in the form of a text in one language (source language) will be analysed by a simple parser. Each of these words will be substituted by a word in another language (target language). There are two approaches of indirect architecture; interlingua approach and transfer approach. In interlingua approach, the input text will be analysed and changed into some abstract representation which is called interlingua. This representation is independent of any language. Then, provided the linguistic knowledge of the target language is known, the *interlingua* can be transformed to the target text.

The transfer approach involves three phases as shown in Fig. 1. The first phase is called analysis. Input text from the source language will be morphologically and syntactically analysed to form lexical units and meaning representation of the source text. The lexical units and meaning representation of the source language will be mapped on to the lexical units and meaning representation of the target language in the second phase. This second phase is called transferring. Finally, in the final phase, the target text will be constructed by synthesizing the lexical units and meaning representation of the target language.

Although the machine translation is normally used for translating text from one natural language into another natural language, it can be also be used for translating text from a natural language into a formal notation. The general approach that have been used in this type of translation can be divided into two types[8]: knowledge-based approach and natural language processing approach. Knowledge based approach uses domain knowledge as the basis in analysing requirements for producing formal specifications. Examples of tools that were developed by using this approach are SPECIFIER [5] and Requirement Analysis[6]. Natural language processing approach combines the knowledge of linguistics and computer sciences. Examples of tools that produce formal specifications from natural language software specification by using this approach are NL2ACTL[7] and FORSEN[8].

DESIGN OF M2Z

M2Z has been designed and developed to translate a software specification statement that is written in Malay into a formal specification statement in Z. In this section, we discuss the design of M2Z.

Overview of M2Z: M2Z uses natural language processing technique in order to extract information from Malay language. Nouns represent sets or types, verbs represent relation between the sets whilst adjectives represent modifiers.

If Saeki *et al.*[13] used an object approach in manipulating and formalising informal requirements for software specification, this study adopts the concepts of entity relationship module for developing the prototype of M2Z as has been done by Vadera and Meziane in FORSEN[8]. The approach used by M2Z is shown in Fig. 4. M2Z architecture uses indirect approach that is by implementing the knowledge of source language and target language.

Every sentence will be analysed by using the rules of the source language grammar (i.e., the Malay language) as well as a database, which consists of five dictionaries. These are stop words[14], nouns, verbs, adjectives and

Table 2: Grammar Rule

Rule	Semantic Interpretation
A → FN FK	FN* (FK*)
FK → KK FN	KK* (FN*)
FK → KK	KK*
FN → KN	KN*
FA → KA	KA*

0	∀	Symbol
1	pelajar	Noun
2	|	Symbol
3	belajar	Verb
4	UKM	Noun
5	•	Symbol
6	mengambil	Verb
7	SKP2210	Noun

Fig. 3: Example of intermediate product

dictionary of mathematical symbols. The result of the analysis is a conceptual model which will then be used by the production for synthesizing predicates by using the knowledge of syntax of Z.

The Z specification language: The formal specification notation Z has been developed by the Programming Research Group at Oxford University over the past decade. We have chosen Z because it is now being accepted as one of the most popular formal specification notation, and it has been applied to a variety of real-life problems.

A Z formal specification document consists of formal passages and informal English explanation. Formal passages in Z is describe by using two complementary languages: the mathematical language and the schema language. The mathematical language of Z is based on set theory. In fact, specifications in Z describe sets, and its construct have their meaning in operations on sets. The schema language supports a structured, systematic presentation of a large-scale specification written in the mathematical language.

Input: The input of M2Z is a statement written in a structured Malay language. The grammar for this language is relatively easier than English. A structured Malay language is the Malay language with restricted vocabulary and grammar. Part of the grammar for this structured Malay language can be represented by using production rules as follows:

<Sentence> → <Noun Phrase> <Verb Phrase>
<Noun Phrase> → <Verb> <Noun Phrase>
<Verb Phrase> → <Verb>
<Noun Phrase> → <Noun>

In order to carry out the study, some assumptions have been made:

- Source text is grammatically correct and meaningful.
- User is familiar with the writing of software specification.

Apart from the text to be translated, M2Z must also be supplied with some information about the model of the system to be specified. For example, basic data types that are used in the model must be given. For example, in the Internal Telephone Directory problem[6], the basic data types involved are

[PERSON, PHONE]

In addition, the function types are relations between PERSON and PHONE, which is declared as follows:

dir: PERSON × PHONE

Both information should be entered before the input text.

Analysis: The analysis phase contains two steps: the lexical analysis and the semantic analysis.

Lexical analysis: The role of lexical analyser is to read every word in the sentence and then to categorize them into one of the classes below:

- Verbs
- Nouns
- Adjectives
- Symbols
- Stop words

Symbols are defined as words that are listed in the translation dictionary of Malay-mathematical symbols. For example, '*setiap*' (English: every) and *'semua'* (English: *for all*) represent universal quantifier, ∀, while *'sesetengah'* (English: *for some*) and '*terdapat'* (English: *there exists*) represent existential quantifier, ∃.

Stop words are words that frequently occur in a text without having any meanings[14]. Words in this class will be removed. However, in this study, words that fall in both stop words class and in symbols class will be classified as symbols. For example, '*yang*' (English: such that) and *'mesti'* (English: *must*) represents symbol | and •.

If M2Z fails to classify the class of a word, it will then prompt the user to helps in classifying that particular word).

<id item>	<argument 1>	<agrument 2>
0	1	1001
2	3	1001
5	6	1001
3	1	4
6	1	7

Fig. 4: Conceptual model based on Intermediate product in Fig. 3

The output from lexical analysis is called an intermediate product. It consists of a table with the following format:

<word_location> <item> <category>

<word_location> is an integer, which represents an index of an array structure; <item> represents the actual word in the sentence and <category> represents the class of the word. For example, the sentence

'setiap pelajar yang belajar di UKM mesti mengambil SKP2201'

(which means: all students that are studying at UKM must register for SKP2201), will be represented by an intermediate product as shown in Fig. 3.

Semantic interpretation: There are several approaches that can be used in interpreting a sentence. A suitable approach is said to be the one that can fulfill the objective of the research as well as the time that is allocated to carry out the research[16]. In this study, semantic interpretation is carried out to identify the relation among entities in a Malay sentence. Therefore, modification approach is used. In this approach, the meaning of a sentence is expressed in its sub-component. For example:

Syntax rule: A → ABC
Semantic rule: sem(A) = function (sem(A), sem(B), sem(C))

Table 2 shows grammar rule and the use of modification approach in interpreting its semantic. The symbol * shows the semantic translation for the rule.

Nouns in an input text form entities. Ordinary nouns represent the name of sets of objects. Whereas, proper noun represents the element of the set[8]. For example, this sentence

'Sesetengah pensyarah mengajar dua kursus'

(which means: Some lecturers teach two courses), has *pensyarah* and *kursus* as its entity whilst in this sentence:

'Setiap pelajar dikehendaki mengambil kursus Teknologi Maklumat'

(which means: all students are required to register for Information Technology course) the proper noun *Teknologi Maklumat* is an element of the entity set *kursus*.

Verbs in input text are identified as relationship among entities. For example, in the first sentence above, the verb *mengajar* relates entity *pensyarah* and *kursus*. Since a relationship can be represented as a set, then the statement can be translated into Z as

$(x, y) \in mengajar$

where x and y are variables of types *pensyarah* and *kursus*, respectively. x and y are arguments or are called arity to the predicate. Non-transitive verb will have a single arity as in the following example:

'Dia tidur.'

(which means: He is sleeping) will produce a Z statement as:

$x \in tidur$

Adjectives that describe nouns will form a predicate with single arity. For example:

'Kursus itu sukar.'

(which means: The course is difficult) can be written in Z as

$x \in sukar$

Besides nouns, verbs and adjectives, this study also considers quantifier in the input text. Quantifier that occurs in input text will be directly translated to mathematical symbol by using dictionary of glossary named *istilah.txt*. This study assumes that input text contains unambiguous quantifier such as the occurrences of numeral words like *'setiap', 'sebahagian'* and such other numeral words in Malay language. For example:

'Setiap kursus adalah senang'

(which means: every course is easy) is translated to:

$(\forall x)\ (senang(x))$.

The following is the semantic interpretation for phrases (FN) that contain both noun and quantifier:

Table 3: Format of predicate and its example

Predicate format	Example
1. [? Quantfier] [? Declaration] • [? Predicate]	$\forall$ y:Z • $y^2 > x$
2. [? Quantifier] [? Declaration] | [Predicate] • [? Predicate]	$\exists$ n:N ، n ≤ 10 • $n^2 = 64$

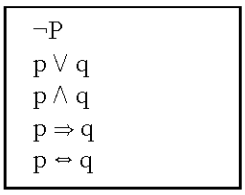

Fig. 5: Position of logical operator in predicates

M2Z ¯ Translator from toe Malay Language into Z

==========

System Model
List tie basic types: (Type "TAMAT" to end)
? pensyarah
? kursus
? tamat

Fig. 6: Screen for input basic types

List the relation: (Type "TAMAT" to end)
Name of the relation ? AjarKursus
Domain ? pensyarah
Range ? kursus
Name of the relation ? tamat

Fig. 7: Screen to input relational types

Type the sentence tobe translated (press "ENTER" to denote the end of the sentence)
→ Terdapat pensyarah yang mengajar [kursus TK1913]

Fig. 8: Screen to input specification sentences

FN → PS KN	$(\forall x)$ (KN*(x))
FN → PK KN	$(\exists x)$ (KN*(x))

where PS is a variable for the universal quantifier translation, PK is a variable for existential quantifier translation, KN is a noun and x is a parameter.

Input text might also contain propositions such as *'dan'* (English: *and*) and *'atau'* (English: *or)*. For example:

'kursus dan pelajar'

is translated to:

kursus ∧ pelajar

Semantic interpretations for phrases that contain prepositions (FK) are as below:

FK → FK dan FK	λx. FK*(x) ∧ FK*(x)
FK → FK atau FK	λx. FK* (x) ∨ FK*(x)

where symbol 'λ' is a lambda calculus and '∧' and '∨' are mathematical symbols. Lambda calculus is used as a formalisme for predicates[16].

Conceptual Model: Conceptual model which result from semantic interpretation is as follows:

<id_item> <argument_1> <argument_2>

where id_item, argument_1 and argument_2 are integers that refer to the location of the word in intermediate product. id_item is either item symbol, verb or adjective. Whilst argument_1 and argument_2 are either noun, verb or adjective. Conceptual model for intermediate product example in Fig. 3 is as in Fig. 4. Integer 1001 indicates that the respective id-item only have one argument.

Production: Production phase is a process to produce Z specification statement from conceptual model. This phase consists of three tasks. The first task is to determine the possible form of type expression of a model. The expression form that is used in this study is limited to:

name or
'(' name 'X' name ')'
'\powerset' name or
'\powerset' '(' name 'X' name ')'

where *name* is a name of data type in the system model. The information used in this phase are obtained from the given basic information of the system model.

The second task is to generate the declaration. In Z, there are two kinds of declaration: basic declaration and compound declaration. The format for basic declaration that is used in this study is as follows:

name {',' name} ':' type_expression

where *name* is a variable name for a set of object and symbol '{ }' shows that several variables with the same type can be declared in a line. Whereas compound declaration is a combination of several basic declaration where each of them are separated by symbol ';'.

The third task is to extract predicates from conceptual model by manipulating logical symbols in the model. Fig. 5 shows the position of logical operator where *p* and *q* are predicates.

00Table 4: Test Result

Subject\Question	Q1	Q2	Q3	Q4	Q5	Q6
M2Z	✓	✓	✓	✓	✓	x
B1	✓	x	x	x	x	x
B2	✓	x	x	x	✓	x
B3	✓	✓	✓	✓	x	x
B4	✓	x	x	x	x	x
B5	✓	x	x	x	x	x
B6	✓	x	x	x	x	x
S1	✓	x	x	x	?	x
S2	✓	x	x	x	x	x
S3	✓	x	x	x	x	x
S4	✓	x	x	x	x	x
L1	✓	x	x	x	x	x
L2	✓	x	x	x	x	x
L3	✓	x	x	x	x	x
L4	✓	x	x	x	x	x
L5	✓	x	x	x	x	x

```
>>>>> The sentence is valid.
>>>>> "outpput. tex" and "output. Dvi" were
successfully generated.
>>>>> The output of the translтion:
\exit a: PENSYARAH | kursus TK1913 \in FKURSUS \\
\t1 \bullet (a, kursus TK1913) \in ajarKursus
>>>>> End of translation.
```

Fig. 9: Screen displays result of translation

The result from the whole proces is an output text which is in the form of predicates. The occurrences of bar symbol ('|') in a conceptual model is important as it will be used to determine the predicate format of the output. The predicate will follow the format as shown in Table 3. In the table, symbol '?' represents the slot that needs to be filled.

Based on the conceptual model in Fig. 5, theZ specifiation statement that will be produced in this phase is:

$$\forall p{:}pelajar \mid (p, UKM) \in belajar \bullet (p, SKP2201) \in AmbilKursus$$

IMPLEMENTATION

M2Z has been implemented by using the C programming language. LEX is used to produce lexical analyzer while YACC is used to generate semantic interpreter.

In this section, some of the screens of M2Z will be shown. Since the interface for M2Z is provided both in Malay and English, we are going to shown the version in English. However, all the input sentences will be in the Malay language.

Fig. 6 shows the first screen of M2Z. User is asked to input about basic information of the respective model. Assume that two basic types are needed that are: *PENSYARAH* and *KURSUS*. A question marked are shown after the user has typed the first basic type that is *PENSYARAH*. In the Fig., another basic type that is KURSUS has been typed. If there is no more basic type to be input, the word *TAMAT* (English: *end*) needs to be written.

The screen in Fig. 7 shows how the user is asked to write the relational types of the model. The user needs to give the relation name, followed by the domain as well as the range of the relation. In the Fig., the name *ajarKursus* has been typed. The user can choose the domain and range by typing the number that with respect (berpadanan) to the available basic types. In this example, number *1* and number *2* have been typed in the respective domain and range. Again, after all the relations have been input, the user needs to type the word *TAMAT*.

Figure 8 shows the screen that received input for specification sentences. The sentence is *terdapat pensyarah yang mengajar [kursus TK1913]*. The square bracket "[" and "]" are used to mark that the word is a special noun. M2Z will omit all the symbols such as comma, dot and such others.

Fig. 9 shows the result of the translation. The output of translation is displayed by following the LaTeX format. Besides that, two output files that are prepared in LaTeX and DVI format are also generated.

If there is syntax or semantic errors, the translation process will be terminated. For every unrecognised words, M2Z will ask the user to identify its word class.

EVALUATION OF M2Z

M2Z is still in the experimentation stage. However, it has shown its ability to translate simple sentence better than most of our students who took an introductory course in formal specification. To test the performance of M2Z, we have selected six sentences. We have asked M2Z to translate these sentences into Z. We have also selected 15 student to do the same task. The list of sentences are as follows:

- Semua pelajar mesti mengambil kursus A (English: All students must take Course A)
- Setiap pensyarah perlu mengajar sekurang-kurangnya 2 kursus (English: Every lecturer must teach at least 2 courses)

  Given:
- 30 pelajar telah mengambil kursus A (English: 30 students have enrolled course A)
- Terdapat pensyarah yang mengajar 2 kursus sahaja (English: There are two lecturers who teach only 2 courses)

Ali boleh mengambil kursus A atau kursus B manakala Ahmad mesti mengambil kursus A dan B

Ali ∈ PELAJAR
Ahmad ∈ PELAJAR
kursus A ∈ KURSUS
kursus B ∈ KURSUS
ambilKursus: PELAJAR ↔ KURSUS

Translation into Z:

Ali ∈ PELAJAR ∧ Ahmad ∈ PRLAJAR ∧ kursus**A** ∈ KURSUS ∧ kursusB ∈ KURSUS
•((Ali, kursusA) ∈ ambilKursus ∨ (Ali, kursusB) ∈ ambilKursus)
∨ ((Ahmad, kursusA) ∈ ambilKursus ∧ (Ahmad, kursusB) ∈ ambiKursus)

Fig. 10: Specification sentences Q8

- Pelajar tidak boleh mengambil kursus B jika dia tidak mengambil kursus A (English: Students cannot enrol for course B if he has not taken course A)
- Pelajar mesti mengambil kursus A sebelum mengambil kursus B (English: Students cannot enrol for course A before taking course B).

Given the following basic information

pel: PELAJAR
kursusA ∈ KURSUS
ambilKursus: PELAJAR ↔ KURSUS
psy: PENSYARAH
kur: KURSUS
ajarKursus: PENSYARAH ↔ KURSUS

- where PENSYARAH, KURSUS and PELAJAR are basic types.

Figure 10 shows one of the sentence that was used in the testing and it translation into Z.

The result of the translation by M2Z and the students are checked to determine the accurary. Table 4 shows the result of M2Z and the students.

All of the students are able to translate the first sentence. However, most of the students are unable to translate the other sentences correctly.

The best performance of the student is given by student B3. This student was able to translate the first four sentences. However he was inccorectly translate sentence number 5 and 6 which are compound sentences. This shows that translating compound sentences is difficult, not just for the computer, but as well as humans.

CONCLUSION

In this study we have described the design and implementation of M2Z, a tool to translate software specification statement that is written in the Malay language to formal specification in Z. The study has shown that it is a meeting point between software specification that is written in a natural language and the formal specification. The lexical and the semantical structures of the words that are used in the informal specification are similar to the structures of the software components in the systems[13].

M2Z is designed to be integrated into WYSIWYG-based Z Editor namely, ZEdit[17]. In order to make it compatible with most of Z tools, the text that is generated by M2Z is presented in Latex form.

In the present form, the tool suffers from a number of limitations. The natural language statement that is accepted by the tool is restricted to a certain grammar rule. Similarly, the formal specification statement is limited to a certain data type. Hence, it is not yet ready to be applied in real example. We are in the process of improving the design of the tool by extending the grammar rule and the range of type of data that can be supported.

ACKNOWLEDGEMENT

We would like to thank the Malaysian Government for sponsoring this project under the IRPA scheme 04-02-02-0039EA106.

REFERENCES

1. Boehm, B.W., 1979. Software Engineering: R and D Trends and Defence Needs. In Research directions in Software Technology. MIT Press., pp: 44-86.

2. Potter, B., J. Sinclair and D. Till, 1996. An Introduction to Formal Specification and Z. 2nd Edn. Prentice Hall, London.
3. Berger, A.L., P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, J.R. Gillet, J.D. Lafferty, R.L. Mercer, H. Pritnz and L. Ureš, 1994. The Candide System for Machine Translation. Proceedings of the 1994 ARPA Workshop on Human Language Technology.
3. Dill, D. and J. dan Rushby, 1996. Acceptance of Formal Methods: Lesson from Hardware Design. IEEE Computer, 29: 23-24.
4. Holloway, C.M., 1997. Why Engineers should consider formal methods? Proc. IAA/IEEE Digital Avionics Systems Conference, 1: 13-22.
5. Miriyala, K. and M.T. Harandi, 1991. Automatic Derivation of Formal Software Specifications From Informal Descriptions. IEEE Transaction on Software Engin., 17: 1126-1142.
6. Reubenstein, H.B. and R.C. Waters, 1991. The Requirement Apprentice: Automated Assistance for Requirements Acquisition. IEEE Transaction on Software Engin., 17: 226-240
7. Fantechi, A., S. Gnesi, G. Ristori, M. Carenini, M. Vanocchi and P. Moreschni, 1994. Assisting requirement formalization by means of natural language translation. in Formal Methods in System Design, Kluwer Academic Publishers, 4: 243-263.
8. Vadera, S. and F. Meziane, 1994. From English to Formal Specifications. The Computer J., 37: 753-761.
9. Harry, A., 1996. Formal Method Fact File VDM and Z. John Wiley and Sons, Chichester.
9. Lee, B. and B.R. Bryant, 2002. Autmated conversion from requirements documentation to an object-oriented formal specification language. Proceeding of the 2002 ACM symposium on Applied computing.
10. Hahnle, R., K. Johannisson and A. Ranta, 2002. An Authoring Tool for Informal and Formal Requirements Specifications. In ETAPS/FASE-2002: Fundamental Approaches to Software Engineering, Edn. By R.D. Kutsche and H. Weber, Springer, LCNS, pp: 233-248.
11. Arnold, D.J., L. Balkan, S. Meijer, R.L. Humphreys and L. Sadler, 1994. Machine Translation: An Introductory Guide. Blackwells-NCC, London.
12. Hutchins, J., 1995. Reflections on the History and Present State of Machine Translation. Paperwork in MT Summit, Luxembourg.
13. Saeki, M., H. Horai and H. Enomoto, 1989. Software development process from natural language specification. Proceedings of the 11th international conference on Software Engin., pp: 64-73.
14. Saad, S., 1997. Pembangunan and Eksperimen ke atas sistem capaian maklumat Al-Quran dwibahasa berasaskan web. MSc Thesis. Universiti Kebangsaan Malaysia.
15. Diller, A., 1994. Z: An Introduction to Formal Methods. 2nd Edn. John Wiley and Sons, West Sussex, UK.
16. Allen, J., 1995. Natural Language Understanding. 2nd Edn. The Benjamin/Cummings Publishing, Redwood City.
17. Shukur, Z., S. Nantha Kumar and Ainita Ban, 2001. Developing WYSIWYG-based Z Editor with Natural Language Capabilities. Proceedings of the International Conference on Parallel and Distribution Processing Technique and Applications, pp: 135-140.
18. Spivey, J.M., 1992. The Z Notation: a reference manual. 2nd Edn. Prentice Hall International Series in Computer Science. Prentice Hall.