ZEMOT: An Integrated Graphic User Interface Prototype for Z Specification
Mobolaji Olufemi Tenibiaje
Assistant Lecturer, Department of Mathematical and Physical Science, Afe Babalola University Ado-Ekiti, Ekitistate, Nigeria
boteny02@abuad.edu.ng

ABSTRACT

Z Specification is one of the most precised formal specification methods due to its mathematical notation and symbols. However, the non-standard font and special symbols of Z notation is posing challenges in the documentation of Z specification language. This paper presents a solution to such problem. An Integrated Prototype System Support for Z Specification Language” ZEMOT (Z Editor Mobolaji Olufemi Tenibiaje)”. This Z integrated Prototype tool will handle description of Z specification, symbol and Latex accessibility and structured schema declaration. The tool enable users to generate schemas in two basic versions; the Z notations and its Latex Mark Up language. This research will focus on development of a functioning GUI editor for Z specification language.

Keywords: Formal Methods, Z specification, Latex Markup Language, Z Notation, ZEMOT

1. INTRODUCTION

Many researches were undertaken in Z specification language and software was developed in solving issues in Z specification Language, yet the common problems are (1.) Documentation of Z specification in latex and (2.) Accessibility of Z notation symbols still poses difficulties in Z specification. Despite the popularity of Z specification language among the other specification languages on precision and other capabilities. It has always been criticized due to lack of associated tool supports (Nasiri 2009). There is no doubt, one of the major reasons why there is a little use of Z specification in software specification world is because there are few accessible tool in Z specification language (Chih& Holcomb 1997).

Based on the above mentioned problem statements such as documentation of Z specification and accessibility of Z specification notations, there is a need to develop a more convenient and accessible tool to encourage researchers to use Z specification Language in formal Methods.

1.1 Formal Specification

Formal Methods can be broadly defined as tools and notations with formal semantics that support the unambiguous specification of the requirements for a computer system. Bowen (2003) defines formal methods has mathematical approaches to software and system development which support the rigorous specification, design and verification of computer systems. Formal method is based on mathematical theory. Some of formal specification languages widely recognized

- VDM, the Vienna Development Method,
- Event B Method and
- Z specification language.

1.2 Z Specification Language

(Pronounced “ZED”)

Z (ZED) specification is a formal specification language based on set theory and first order predicate logic. Z specification notation is used to precisely specify the behavior of a system and analyse them via proof, animation, test, generation and so on (Miller, 2005).

Mohamed & Zarina (2008) described formal specifications, such as Z is based on mathematics. Though formal specifications are very precise and accurate in specification of software and systems, Z specification has been considered to be more effective in representing software specifications. Z notations are precise and unambiguous and thus the formal notation always provides the definitive description in the case of any misunderstanding (Bowen, 2003). Z specification language was designed for not machine but for man understanding of software specification. Z specification according to Beidler () is a non-executable in general that is, is not a programming language but a specification language. Z specification is needed as in programming languages. Spivey (1988) described Z, as a specification language based on classical set theory. Jacky (2001) describes Z specification as a model-based notation. Z is a notation not a method but support many different methods.

Every Z specification language documentation consist a series of numerous class of Paragraphs. The likes of Axiomatic Operation Schema and Generic schema which are defined by the Basic Type schema.

2. ZEMOT USES AND APPLICATION

ZEMOT is a Z specification tool with the name coin from Z editor From Mobolaji Olufemi Temibiaje. ZEMOT is an application that allows the creation of schema, edit of Z notations and saving the schemas in Latex and notational format files. ZEMOT is Z integrated prototype tool that handles description of Z specification, symbol and Latex accessibility and structured schema declaration, on development of a functioning Graphic User Interface editor for Z specification language. The GUI prototype also allows schema description in basic types, axioms, and schemas (Generic and Operational).
Symbol list is prepared for giving the exclusive use for basic type, schema, and Symbol List are presented. ZEMOT was developed using Visual Basic 2010. Net framework.

The specification of Birthday Notification System description in Z specification is presented here using ZEMOT editor with the axiomatic description and the schema.

2.1 Main Schema Window
The main window displays the SCHEMA, the level of process, BASIC TYPE and the variables used in the course of the specification. Users can specify the Basic type from the main window.

![Figure 1: Main ZEMOT window](image1)

The BASIC type just focuses on essentials and hold off looking at details. The Basic types are made available to all the windows in ZEMOT. Likewise the global variable window builder and schema window. Here the user can specify the “Global Variable” needed in the course of the Specification of system. The Global is accessible to all the windows in the ZEMOT. At the left side of the schema window panel the listed global variable are displayed. Similar every elements of the global variable displayed below the “General Variable”.

![Figure 2: Basic Type Window](image2)

It is convenient to describe Z documents consist of many different components such as Axiom, Generic Constant, Basic Type, and Schema. Moreover, ZEMOT can be saved and exported Z documents not only in LATEX form but also in notation one.

2.2 Basic Type
The above figure is the Basic type window of ZEMOT. The basic types characters used for birthday notification system notation are Bday and Name. The characters are separated with comma.

Figure 3: Notation view and Latex view of the generated Basic type builder

The above figure is the main window showing notation and the Latex Markup Language (LML) format of the specification. The upper portion of the figure describes the Notation format of the Z specification of basic types are described between [ and ] While the Lower panel of the figure indicates the basic type in LML which begins \begin{zed} and ends \end{zed}. Notation and Latex markup Language formats of Z Specification can be saved in ZEMOT.

2.3 Axiomatic Descriptions

An axiomatic description declares the global variables and constants for the entire specification and discretely specifies a constraint of their variables. An Axiomatic description is divided into two parts: Declaration and Predicate. ZEMOT editor provides two entry text area which enables a structured description of axiom with the aid of the Declaration builders and the Predicate builder. Axiom Description Declaration Builder imposes the checking policies of axiomatic description with notation and its Latex equivalent generation. Axiom Description Predicate Builder enforce the checking policies of the Predicate while the notation format and its Latex equivalent generation.

2.4 Generic Schema

Figure 4: Generic schema Builder

ZEMOT Generic Schema Builder provides schema builder in two basic level of paragraph for every builder. The Builders are generic schema declaration builder and predicate Builder windows.

The generic declaration schema builder aids the declaration of the Schema of the generic Schema. The builder enforces the declaration construction principles and only the need symbols for the construction are made available.

The generic predicate schema builder window aids the construction of predicate paragraph of generic Schema. The builder enforces the predicate construction principles and only the need symbols for the construction are made available.

The global variable are accessible and the user can easily make a better and structured declaration schema the predicate builder helps the user to select the needed symbol and the bracket perform schema bracket function. The predicate builder helps in constructing a more structured predicate schema for the generic schema.

2.5 Operation Schema

Figure 5: ZEMOT Operation Schema Builder

ZEMOT operation schema builder provides schema builder in two basic level of paragraph for every
The Builders are operation schema declaration builder and predicate Builder windows.

The operation schema declaration builder helps in the declaration of the Schema of the operation Schema. The operation predicate schema window can access global variables while user can easily make a better and well-structured declaration schema the predicate builder helps the user to select the needed symbol and the bracket perform schema bracket function. The predicate builder window helps in constructing a more structured predicate schema for the operation schema.

![Figure 6: Collections of window within ZEMOT](image)

Examples of the schema generated by ZEMOT are shown below. The builders allow a proper generation and specification of schemas, basic type, state and Axiomatic description is provided below

<table>
<thead>
<tr>
<th>Notation</th>
<th>Latex markup language</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Bday,Name]</td>
<td>\begin{zed} [Bday,Name] \end{zed}</td>
</tr>
<tr>
<td>![blong]</td>
<td>![gendif] + ![blong] bd? : Bday &amp; fun date nam? : \set Name \where unkn? : = dom Name \end{gendif}</td>
</tr>
<tr>
<td>![Addbirthday]</td>
<td>\begin{schema} \delta Addbirthday \end{schema} nam? : \set Name date? : Bday \where nam? : \notin unknow bd' : Bday {nam?\rightarrow date?}</td>
</tr>
<tr>
<td>![find Birthday]</td>
<td>\begin{schema} \xi find Birthday \end{schema} nam? : Name date? : Bday \where nam? : \in unknow date? = Bday {bd?\rightarrow nam? }</td>
</tr>
</tbody>
</table>

3. CONCLUSION

In this paper, researcher presents a Graphics User Interface prototype for Z specification language which allows a more structured and well-formed schema generation. ZEMOT (Z editor Mobolaji Olufemi Tenibiaje) encourages structured and well-formed schema by providing the needed symbols to a specific schema windows of the system. In order to avoid wrong use of symbols, the syntax checking functionality is integrated within the schema builders while the system has been structured to eliminate the wrong use of symbols in schema generation.

ZEMOT allows the user to focus on working with the notations and not been distracted with the need to learn Latex commands and tools. Meanwhile ZEMOT allow the user to understand the Latex command equivalent of the every Z notation symbols.
ZEMOT is an integrated prototype tool for Z specification. ZEMOT was successfully developed based on the documentation problem of Z specification, especially researchers in carrying out specification in Z specification. This research was carried out to provide aiding tools for researchers to specify software in Z specification language and also aiding easy documentation of schema of Z notation in Latex markup language (LML)

REFERENCES


AUTHOR PROFILE
Mobolaji Olufemi Tenibiaje received his first degree in Computer science and Education from University of Ado- Ekiti, Ekiti State, Nigeria, in 2008. He received his MSc in Computer Systems Engineering (Software Options) from University of East London in 2012. Currently, he is an Assistant Lecturer at Afe Babalola University Ado- Ekiti.