University of Ibadan
Journal of Science and Logics in ICT Research

Design of an African Native Language-based Programming Language

Ezekiel K. Olatunji[1] aekolatunji@gmail.com
John. B. Oladosu[1] johnoladosu@gmail.com
Odetunji A. Odejobi[2] oodejobi@yahoo.com
Stephen O. Olabiyisi[1] soolabiyisi@lautech.edu.ng

[1]Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria
[2]Department of Computer Science and Engineering, Obafemi Awolowo University, Ile-ife, Nigeria

Abstract

There is scarcity of research information on programming languages developed with the lexicons of an African indigenous language unlike those with the lexicons of European and Asian languages. This study explored the design of an African native language-based programming language using Yoruba as case study. Yoruba is the first language of over 30 million people in the south-west of Nigeria, Africa; and is spoken by over one hundred million people world-wide. It is opined, as confirmed by research studies, that making computer programming possible in one's mother tongue will enhance computer-based problem-solving processes by indigenous learners and teachers. Furthermore, such a feat will empower those only literate in their mother tongue to program or learn programming of the computer in their mother tongue. In addition, successful implementation and adoption of such a programming language will increase the functional load of the base language, thus reducing the chances of its natural extinction as has been predicted for most African languages. The character set and reserved words of the programming language were respectively formed from the basic Yoruba alphabets and standard Yoruba words. The lexical items of the programming language were specified by defining necessary regular grammars while appropriate context-free grammars were defined, using Backus-Naur Form (BNF) notations, to specify the syntactic structure of a valid program and program elements. As a prelude to implementation of the designed programming language a mini scanner was developed using QBASIC within QB64 integrated development environment (IDE). A subset of the syntactic definitions of the programming language was produced as the primary output. In addition, a token recognizer for the designed programming language has been produced. The output produced by the developed token recognizer showed its functional correctness

Keywords-Programming languages, Lexicons, Yoruba language, BNF, Token recognizer

I. INTRODUCTION

Programming languages (PL) are notations for description of algorithms and data structures to computers and people. Evolution of programming languages (PL) and their implementation have always been influenced by a number of factors including discovery of weaknesses and/or deficiencies in the existing PLs; development in computer hardware; requirements of new areas of application; changing understanding of better methods of writing and maintaining large and complex programs; understanding of strength and weakness of some language features and the need for standardization [1,2].

Ezekiel K. Olatunji, John. B. Oladosu, Odetunji A. Odejobi and Stephen O. Olabiyisi (2019). Design of an African Native Language-based Programming Language, *University of Ibadan Journal of Science and Logics in ICT Research (UIJSLICTR)*, Vol. 3, No. 1, pp. 72 - 78

Most of the existing popular high-level programming languages (such as Pascal, Visual BASIC, Java, C++ and others), especially in the continent of Africa, borrowed their lexical items from English language. There are also literatures on PLs that borrowed their lexical items from Asian languages (such as Rapira [3] and Ezhil [4]). However, there is scarcity of research information on serious efforts at designing and implementing PLs based on the lexicons of African indigenous languages, such as Yoruba language.

According to [2], development of such programming languages will improve computer-based problem-solving processes by indigenous teachers and learners. This fact has also been corroborated by research studies [5, 6, 7, 8]. This research explored the design of an African native language-based programming language using Yoruba as case study. Yoruba is the first language of over 30 million people in the south-west of

Nigeria, Africa; and is spoken by over one hundred million people world-wide [9].

The need for African native language-based programming languages cannot be over emphasized. According to Alidou *et. al.* [10], the African continent is the home of about one-third of the world's living languages; and yet there are many people, in the order of millions, who are only literate in their mother tongue (in the sense of not being proficient in both spoken and written English language). Availability of a mother tongue-based programming language will enable many millions in Africa who are only literate in their mother tongue to learn how to program the computer. There are over 13 million of this category of people in Nigeria alone by 2010 [11]. Furthermore, such a programming language, when developed, will increase the functional load of its base language, which will consequently reduce the chances of such languages going into extinction as predicted and feared in many circles [12, 13, 14].

II. LITERATURE REVIEW

While there is dearth of literatures on African native language-based Programming languages, there exists some PLs that are not based on English words but are based on the lexicons of other native languages. Most of these programming languages were also developed for similar reasons as our designed Yoruba-based PL – the need to simplify learning and teaching of computer programming by indigenous people as well as to empower millions of people in Africa who are only literate in their mother tongues.

The development of Qalb, a functional programming language based on the Arabic language was reported by Nasser [15]. One of the motivations for developing Qalb was to challenge the culture in which the design of most modern popular PL is predominantly based on the English language words. Such a culture, as noted by McAllister [16], makes learning programming difficult especially for students whose native language does not even use the Latin alphabets as does English language. While Qalb's syntax is similar to that of Lisp and Scheme, the syntax of our programming language is similar to that of structured BASIC. Furthermore, unlike Qalb which did not make use of ASCII character set for its encoding [15], our PL made use of a mixture of ASCII and Unicode characters.

Kumar [17] described the development of the Hindawi programming system (HPS), a suite that allows users to program in Indic languages (Hindi, Bangla, Gujarati, Assamese and some other Indic languages). The HPS, developed by Chaubary, A and Chaudbary, S., is a free, open-source, completely non-English-based programming platform that allows non-English medium literates of India to learn and write computer programs [18]. The HPS removes the English language barrier and enables non-English literates Indians to take up computer science and participates in the information and communications technology (ICT) at all levels of technology from primary school education to robotics and super-computing in their mother tongue. While their work provided a programming platform for many indic languages, our is just one African language – the Yoruba language. However, the planned implementation approach for our work is similar to that of Chaudbary [18].

"Dolittle", developed by Kanemune and Kuno[19], is a PL that is based on the lexicons of Japanese language. The language was developed in 2000 [20] in response to the need for an object-oriented programming language that is suitable for children in both elementary and secondary schools in Japan. It was designed to be an object-oriented educational programming language. The programming language was written in Java and does not require elaborate declarations. Unlike the work of Kanuemue and Kuno [19], the keywords and identifiers in this research are based on Yoruba language. Furthermore, structured programming is the focus of our designed language.

Ezhil is an interpreted Tamil-based programming language developed by Annamalai in 2009 [21]. Tamil is an Indian language spoken by over 60 million people [22]. The programming language is targeted towards the K-12 (Junior high school) level Tamil speaking students as an early introduction to thinking like a computer scientist [4]. The syntax of Ezhil is broadly similar to that of conventional BASIC. According to the author of Ezhil, the primary motivation for developing the language is that "like mathematics, computing is a concept and can be introduced through any native language". Thus by introducing computing in the native language children can easily learn how to think in the required "logical modes (enumeration, recursion, procedural)" [21]. The syntax of our PL is similar to structured BASIC, and only Yoruba keywords are allowed.

A. Theoretical Framework of the Research

This research derives its theoretical underpinning from the theories of formal grammars, languages

and automata. The Backus – Naur Form (BNF) notation and its extended version introduced by John Backus and Peter Naur in 1960 as, cited by [23], are used in describing the syntax of the Yoruba-based programming language. According to Chomsky hierarchy of grammars and languages [24], the Yoruba-based PL belongs to the class of type-2 language, which is defined by type-2 (context-free) grammar. The lexical structure of the PL was defined by type-3 (regular) grammar. Furthermore, with respect to automata theory, as propounded by Alan Turing [25], valid sentences in our Yoruba-based PL can be recognized by pushdown automaton, while its lexical items can be recognized by a finite state automaton (FSA).

B. Design goals of the Programming Language

The key design goal of the PL is to make use of Yoruba language lexicons for the reserved word and identifiers. The programming language is designed to be pedagogically simple, even for children in senior primary and junior secondary schools. The lexicons of the base language (Yoruba), particularly the key/reserved words and identifiers are, however, diatric-free; that is, they have no tone markings. This is because programming language is concerned with communication with computer in a written form and not in a spoken form.

III. METHODOLOGY

This work is on the design of a programming language and not on the implementation of the language. While design of a PL refers to the specification of the lexical and syntactic structures of the language, implementation refers to development of a processor (such as a compiler or interpreter) for the language. The specification of the lexical structure of the Yoruba-based PL was accomplished by defining necessary regular (type-3) grammar. The character set of the language was formed from the basic Yoruba alphabets. Some standard Yoruba words (such as 'Sesiro', "Gbawole" and others) that will not lead to ambiguity were used for the design of the lexical items of the PL

Furthermore, appropriate type-2 (context-free) grammars were designed to specify the syntactic structures of a valid program and program elements (statements, expression, and others) in the language using Backus-Naur Form (BNF) notations and its extended version (EBFN). The use of the BNF notation is the preferred approach to designing the syntactic structure of a language as opposed to using syntax diagrams.

While the core of the research is on design of the Yoruba-based PL, a mini token recognizer was developed in QBasic 64 within QB64 integrated development environment (IDE) as a prelude to its full implementation, The PL is intended to be implemented as a source-to-source compiler as was for early versions of Java, Python and others (26).

IV. RESULTS AND DISCUSSIONS

A. Result of the Designed PL

The primary output of the design of the Yoruba-based PL is its syntactic definitions. These are presented and described in this section. A subset of the output, for want of space, is as provided in Figures 1, 2 and 3. The Syntactic element <PurogiramuYoruba> is the start symbol of the grammar for the Yoruba-based programming language. Other non-terminal symbols, all of which are enclosed in corner bracket, such as <PurogiramuYoruba>, are as provided in the production rules of the grammar for the language. The terminal symbols are the character set (the derivatives of the non-terminal symbol <Dijiti> and <ABD> in figure 3) and reserved words (such as "Sesiro", "Gbawole" and others). These are not enclosed in a corner bracket. The production rules are as given in the detail definitions of the language, part of which are provided in Figures 1, 2 and 3.

Figure 1 contains the production rules for the start symbol, <YorubaPurogiramu>, in the language as well as for, <Dikilaretifu> and <Awon-Oroo> non-terminals symbols. The production rule for the start symbol means that a valid program in the language must begin with the word 'Ibẹrẹ' followed by non-terminal <Dikilaretifu>, then by non-terminal symbol <Awon-Oroo> and ends with the terminal symbol, the word 'Opari'. This is how to interprete each production rule. Shown in Figure 2 are the production rules for < Osesiro>,< Bibẹni > and <TabiBẹkọ> non-terminal symbols as well as their subordinate non-terminal symbols. The production rules for non-terminal symbols <Nọmba> and < Feriebu> are shown in Figure 3. In Figure 3, the production rules for <Dijiti> mean a digit can be any of numeral zeros to nine.

The lexical items of the designed PL include reserved words, identifiers (which are represented by the syntactic entity <Feriabu> in Figure 3), integer numbers (represented by syntactic entity

<Nomba> in Figure 3), string constants and delimiters (including some familiar arithmetic operators and relational operators). Furthermore, there are four context sensitive requirements of the programming language. First, variable names must be declared before they are used and a variable name may only be declared once. Furthermore, a program must begin with the 'Ibẹrẹ' statement and terminates with the 'Opari' statement.

B. Result of Part Implementation of the PL

The result of part implementation of the programming language is the development of a scanner for the language. The scanner was tested with a simple source program in Yoruba shown in Figure 4. Figure 5 is a sample output for source statement 4 in Figure 4, while Figure 6 is the scanner output for the entire test data as will be seen by a programmer. Figure 5 contains the different tokens recognized in the source statement 4 along with their token categories.

In the test data, exclamatory mark (!) precedes a comment statement, while a string literal is enclosed within two single quotes. Figure 6 shows that characters 'Q' and 'Z' in statement 4 of Figure 4 are correctly recognized as being invalid characters in the alphabets of the designed Yoruba-based programming language.

Definition of a Yoruba program having <PurogiramuYoruba> as the Start Symbol

<PurogiramuYoruba> → Ibẹrẹ <Dikilaretifu><Awon-Qrọọ > Opari

Definition of <Dikilaretifu> (Declarative statement)

<Dikilaretifu> → Intija < IẸsipẹrẹ > | Karakita < IẸsipẹrẹ>
< IẸsipẹrẹ> → <Orukọ> | < Orukọ> , < IẸsipẹrẹ>
This could as well be written as:
< Dikilaretifu> → (Intija | Karakita) <IẸsipẹrẹ>
<IẸsipẹrẹ> → < Orukọ> [, <IẸsipẹrẹ>]

Definition of < Awon-Qrọọ> (Statement-Sequence)

< Awon-Qrọọ> → <Qrọọ> | <Qrọọ>< Awon-Qrọọ>
< Qrọọ> → <OKọjade > | <OGbawọle > | < OJẹki >| <Osesiro> |
<Bibẹni> | <TabiBẹkọ> | <OseNigbati> | <Alaayee>

Figure 1: the production rules for the start symbol

Definition of < Osesiro> (Arithmetic statement)

< Osesiro> → Sesiro < Feriebu> = <IsiroẸsipẹrẹ>
< IsiroẸsipẹrẹ> → < IsiroẸsipẹrẹ> + <Taamu> | < IsiroẸsipẹrẹ> - < Taamu> |
<Taamu>
< Taamu> → < Taamu> * <Fakitọ> | < Taamu> / < Fakitọ> | < Fakitọ>
< Fakitọ> → < Feriebu > | < Nọmba> | (<IsiroẸsipẹrẹ>)

Definition of < Bibẹni > (if-statement) and <TabiBẹkọ> (if-Else-statement)

< Bibẹni > → Tobajẹpe <ẸsipẹrẹAfiwe> Nigbana < Awon-Qrọọ> Tobajẹ-pari
<TabiBẹkọ> → Tobajẹpe<ẸsipẹrẹAfiwe>Nigbana<Awon-Qrọọ>Bibẹkọ<Awon-Qrọọ>
Tobajẹ-pari
< ẸsipẹrẹAfiwe> → < IsiroẸsipẹrẹ ><IseAfiwe>< IsiroẸsipẹrẹ>
<IseAfiwe> → = | < | >

Figure 2 are the production rules for Osesiro

Definition of < Nọmba> (Number) and < Feriebu> (Variable)

< Nọmba> → <Dijiti> [{ <Dijiti> }^{4}]
< Feriebu> → <ABD> [{ <ABD> | <Dijiti>}^{6}]
<Dijiti> → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<ABD> → A | B | D | E | Ẹ | F | G | 'GB' | H | I | J | K | L | M | N |
O | Ọ | P | R | S | Ṣ | T | U | W | Y |
a | b | d | e | ẹ | f | g | 'gb' | h | i | j | k | l | m | n | o | ọ | p | r | s | ṣ |
t | u | w | y

Note:
The definition of <Nọmba> implies it cannot be greater than 99999, while a
< Feriebu> which is an identifier cannot be more than 7 characters long.

Figure 3: Production Rules for <Nomba> and <Feriabu>

```
Debo = Ade
Ade = 5
Abd = 4 (+
Jeki abd45yu = 56W  / Q4 '- ) ihjkl + 'mnop'  * ! rst < u1 w
!Abd = 4 (E+ma) >Z - Ghogo / Q2
Abd = 4 (E+ma) >Z - Ghogo / Q2
```

Figure 4: Source Program in Yoruba for Testing the Scanner

```
Statement  4  : Jeki abd45yu = 56W  / Q4 '- ) ihjkl + 'mnop'  * ! rst < u1 w
S/No Actual Token   Token category/Internal Repr
************************************************************
 1   Jeki           Reserved word
 2   abd45yu        Identifier
 3   =              Operator
 4   56             Integer
 5   W              Identifier
 6   /              Operator
 7   4              Integer
 8   - ) ihjkl +    Litira Olooro
 9   mnop           Identifier
Statement  4  : Jeki abd45yu = 56W  / Q4 '- ) ihjkl + 'mnop'  * ! rst < u1 w  (
60 )  Has been processed
Press any key to continue_
```

Figure 5: Scanner's Output for Statement 4 in the Source Program

```
Scanner Produced Comments & Error File
==================================
Asise ni ila   4  : Q je Karakita Ajeji
Asise ni ila   4   * ! rst < u1 w  Literal Oloroo yi   kodara to
o
Oro Alaye ni ila  5  : !Abd = 4 (E+ma) >Z - Ghogo / Q2
Asise ni ila   6  : Z je Karakita Ajeji
Asise ni ila   6  : Q je Karakita Ajeji
```

Figure 6: Scanner's Output for the entire source program as will be seen by the Programmer

V CONCLUSION

The need to facilitate better comprehension of problem-solving process using computer by indigenous learners and teachers, as confirmed by research studies, is a major reason for embarking on research in development of indigenous language-based programming languages. In addition, the need to empower more people who are only literate in their native languages is an additional worthwhile reason for attempting the design and implementation of a mother tongue-based programming language. Availability of such programming languages will serve as an effective strategy to:

i) bridge the digital divide between the developed and under-developed/developing countries of the world [6, 27].
ii) ensure the continuity and relevance of indigenous languages in the age of ICT and increasing globalization
iii) prevent predicted linguistic genocide of indigenous African languages [12, 13].

The next stage in this research is to embark on full implementation of the PL by developing a suitable source-to-source compiler for the language.

REFERENCES

[1] Olatunji, E. K (2014): Programming Languages – Introductory Text on Concepts and Principles, Ilorin – Nigeria, Igbagbogbemi Publishers ISBN 978-187-458-9, pp 8.

[2] Olatunji, E.K., Oladosu, J.B., Odejobi, O. A and Olabiyisi, S.K.(2018), Towards Development of an Indigenous African Language-based Programmining Language, FUOYE, Journal of Engineering and Technology (FUOYEJET).

[3] Dadhich, B.S. (2006), Needed:A good 'Hindawi Programming Language'.[Online] Retrieved on 15-06-2015 from www.localisationlabs.balendu.com/

[4] Annamalia, M. (2013), Invitation to Ezhil: A Tamil Programming Language for Early Computer Science Rducation. [Online] Retrieved from http://ezhillang.org on 17-03-2015

[5] UNESCO (2007), Mother Tongue-based Literacy Programmes: Case Studies of Good Practice in Asia. Bangkok: UNESCO Bangkok, Thailand, ISBN 92-9223-113-8 [Online] Available online at www.unesdoc.unesco.orgRetrieved on 30-07-2014

[6] Pflepsen, A. (2011), Improving Learning Outcomes through Mother Tongue-Based Education, , MTB-MLE Network. [Online] Available online at https://www.eddataglobal.org Last Accessed on 30-07-2014, stored online as eddata_ii_mother_tongue_instruction.pdf Vol 1, No 2, pp 148-159, Dec 2013.

[7] UNICEF (2011). Action research on mother tongue-based bilingual education: Achieving quality, equitable education. [Online] Available online at http://www.vn.one.un.org.Retrieved on 30-07-2014

[8] Pamela J. Mackenne and Jo Walker (2011) Mother-Tongue Education: Policy lessons for Quality and Inclusion, Global Campaign for Education, Johannesburg, South Africa. [Online] Available at www.campaignforeducation.orgRetrieved on 30-0702014

[9] Eludiora, S.I.; Agbeyangi, A.O. and Fatusin, O.I. (2015). Development of English to Yoruba Machine Translation System for Yoruba verbs's Tone Changing, International Journal of Computer Applications, vol. 129, No. 10

[10] Alidou Hassana, AliouBoly, Birgit Brock-Utne, Yaya Satina Diallo, Kathleen Heugh, and H. EkkehardWolff (2006),Optimizing Learning and Education in Africa– the Language Factor, [Online] Available at www.adeanet.orgRetrieved on 30-07-2014.

[11] NBS(2010), The National Literacy Survey. A Publication of National Bureau of Statistics, Abuja, Nigeria [Online] Retrieved on 25-06-2015 from www.nigeriastat.gov.ng

[12] Benson, C. (2006) , "Mother Tongue First". [Online] Available on line at www.multicultures.comRetrieved on 30-07-2014

[13] Azeez, A.S (2013), The Yoruba language and Literature in the 21stCentuary and beyond, Covenant Journal of language Studies (CJLS), Vol 1, No 2, pp 148-159, Dec 2013.

[14] Awobuluyi, O (2014), Yoruba Must Not Die Out, A Lecture delivered at the Faculty of Arts, Obafemi Awolowo University, Ile-ife, Nigeria, Feb. 2014

[15] Nasser, R. (2012), Qalb, [Online] Retrieved from http://nas.sr/%D9%82%D9%84%D8%A8 on 18-03-2015

[16] McAllister, N. (2013), Meet Qalb, the Programming language that uses Arabic Script. [Online] Retrieved on 18-03-2015 from http://www.theregister.co.uk/2013/01/25/arabic_programming_language

[17] Kumar, L (2011), Hindawi: Write Computer Programs in Indic Langauges, [Online] Retrieved from http://techwelkin.com/hindawi-now-write-computer-programs-in-indic-langaugeson 18-03-2015

[18] Choudbary, S. (2008), The Hindawi Programming System, [Online] Retrived from http://hindawi.in/en.US/ on 17-03-2015

[19] Kanemune, S and kuno, Y (2005), Dolittle: An Object-oriented Language for K12 Education, Warsaw, Eurologo 2005, pp 144-153. [Online] Retrieved fromeurologo2005.oeiik.waw.pl on 18-03-2015

[20] Yoo, S. W., Kim, K., Kim, Y., Yeun, Y., Kanemune, S and Lee, W (2006), Empirical Study of Educational Programming Languages for K-12: Between Dolittle and Visual Basic; in International Journal of Computer Sciences and Networks Security, Vol, No 6, 2006.

[21] Annamalia, M (2009), Ezhil – A Tamil programming language; [Online] Retrieved from arxiv.org/pdf/0907.4960 On 17-03-2015

[22] Subramanian, K. (2015), Bridging the Digital Divide with Tamil Coding. [Online] Retrieved from www.thehindu.com/~/making computing-in-tamil-easy/ on 17-3-2015

[23] Wirth, N. (2005). The Programming Language Pascal. Software Pioneers (pp. 121-148). Springer Berlin Heidelberg

[24] Petersen, W (2006), Introduction to the theory of formal languages. [Online] Retrieved on 08-07-2015 from phil.fak.uni.duesseldorf.de/~/Petersen/

[25] Kari, J (2013), Automata and Formal Languages. [Online] Retrieved on 08-07-2015 from users.utu.fi/jkari/automata/fullnotes.pdf

[26] Scott, M.L. (2009), Programming Language Pragmatics, New York, Elsevier; ISBN 13: 978-0-12-374514-9

[27] Wagner, D. A., C. J. Daswani and RomillaKarnati (2010.)Technology and Mother-Tongue Literacy in Southern India: Impact Studies among Young Children and Out-of-School Youth Creative Commons Attribution-Non Commercial-Share Alike 3.0, Volume 6, Number 4, Winter 2010, 23–43. [Online] Available at www.itidjournal.org Last accessed on 30-07-2014