World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:5, No:3, 2011

Design of Encoding Calculator Software for Huffman and Shannon-Fano Algorithms

Wilson Chanhemo, Henry. R. Mgombelo, Omar F Hamad, and T. Marwala

***Abstract*—**This paper presents a design of source encoding calculator software which applies the two famous algorithms in the field of information theory- the Shannon-Fano and the Huffman schemes. This design helps to easily realize the algorithms without going into a cumbersome, tedious and prone to error manual mechanism of encoding the signals during the transmission. The work describes the design of the software, how it works, comparison with related works, its efficiency, its usefulness in the field of information technology studies and the future prospects of the software to engineers, students, technicians and alike. The designed "Encodia" software has been developed, tested and found to meet the intended requirements. It is expected that this application will help students and teaching staff in their daily doing of information theory related tasks. The process is ongoing to modify this tool so that it can also be more intensely useful in research activities on source coding.

***Keywords*—**Coding techniques, Coding algorithms, Coding efficiency, Encodia, Encoding software.

I. Introduction

SOURCE coding is very important in telecommunications industry as it attempts to compress data by remove redundancy from the source in order to transmit it more efficiently. A good example in this case would be in the internet where the common "Zip" data compression is used to reduce the network load and make files smaller. There are many source coding algorithms which are used depending on the application. These include Shannon-Fano algorithm, Huffman algorithm, Lempel Ziv algorithm, and Morse coding algorithm.The encoding calculator, dubbed Encodia, discussed in this paper is designed to help in realization of source coding separately using Shannon-Fano and Huffman algorithms and analysis of source entropies based on these algorithms. The designed software gives options of the scheme to apply – Huffman or Shannon-Fano or both.A fine question could be "why is Encodia needed?" The need for this software came after the realization of, among many, the following difficulties which students undertaking information theory studies and the tutors teaching the subject do face in their daily assignments by trying to solve some encoding problems manually; - (i) Manual encoding process becomes more difficult as the number of symbols increases; (ii) It is difficult to track any error made in between and, hence, this can lead to wrong answers; (iii) The procedure takes long time to complete; and (iv) It becomes almost impossible to manually realize the algorithms with text samples.An effective and competent solution among the solutions though for the stated problems and difficulties has been found to be a design of automated software which will accept inputs from the user and then perform the required task accordingly.).

II. Related Works

The design of the Encodia is not the first attempt in an effort of source coding realization. There has been a number of related works as per the following discussion.The project Huffman encoding implemented in C was done by Rajiv A Iyer in [6]. It, however, only demonstrates encoding of symbols when their probability of occurrence in a particular text is known, and it is written using C language syntax. It is a command prompt based application. It is limited to a maximum of 64 symbols and the maximum number of bits which can be produced for each symbol is just 100.On the other hand, the Shannon-Fanno encoding project implemented in C was done by Davis [8]. Again, it only demonstrates encoding of symbols with their probabilities of occurrence in a give text is known. The application is command prompt based written in C language. It is limited to 15 symbols only. This is too small for practical usage.Therefore, the need to design a more robust and better symbol accommodating scheme has been unavoidable. We present the proposed Encodia design and its implementation together with its evaluation in comparison to the manual mechanisms of encoding the symbols.

III. Encodia Design

Having analyzed the above and other related works, the task was to design a software which will be: (a) More robust; (b) User friend; (c) Free for any user; and (d) Compact to take as little memory as possible. In order to meet these objectives, Encodia has been designed following two essential approaches. These are Object-Oriented approach with the help of Unified Modeling Language (UML) for analysis and design, and Object Oriented Programming (OOP) use in the implementation part.Object-Oriented approach is used instead of other methods like Procedural Programming because of simplicity, maintainability, re-usability, and extendibility or scalability**.** Adding new features or responding to changing operating environments can be solved by introducing a few new objects and modifying some existing ones.

IV. Specifications of Requirements

The requirement specifications explain the needs for the Encodia so that it can work as planned for. In overview, this work intends to design software for source coding which applies Huffman and Shannon-Fano algorithms.The main design goal being to reduce difficulty in source coding done manually and save time for doing this activity while avoiding

the possibility of introducing human errors, this work has proposed, designed , and implemented a robust, freeware, and compact system.Function-wise, the system is supposed to do what is expected to be done by reacting in particular situations [1]. System functions can be either evident or hidden. It should be performed and the user should be recognizant that is performed or the functions are performed but not visible to users. Consequently, the Encodia software will need the functions as shown in Tables I and II.

TABLE I
EVIDENT FUNCTIONS

Ref#	Function	Category
R4.1	Choose type of input	Evident
R4.2	Choose encoding scheme	Evident
R4.3	Enter probability of symbols	Evident
R4.4	Enter text	Evident
R4.5	Display output	Evident
R4.6	Display error massage due to wrong input	Evident
R4.7	Calculate entropy	Evident
R4.8	Calculate average word length	Evident
R4.9	Calculate efficiency	Evident
R4.10	Compare efficiency	Evident
R4.12	Calculate probability of individual symbols	Evident
R4.13	Display symbol probabilities	Evident

TABLE I I
HIDDEN FUNCTIONS

Ref#	Function	Category
R4.14	Count number of symbols in a text	Hidden
R4.15	Arrange symbols in descending order of probabilities	Hidden

The system attributes, also called non-functional requirements [1], are constraints imposed on the system and restrictions on the freedom of the designer. The system attributes for the designed Encodia are shown in Table III.The software requirements constrain the environment and technology of the designed Encodia software [3]. They include the platform on which it will run and the development tool to be used during programming. The Platform has been Windows with .NET framework 2.0 and above while the design tool has been Microsoft Visual studio 2008 and the programming language is C-sharp (c#). In order to implement a system the choice of a processor with maximum possible speed was made. There should be sufficient memory to store data and software tools for efficient processing. Requirements for the Encodia to work properly are at least: IBM-Compatible PC of Pentium IV and above processor with speed of at least 1.5GHz, memory of at least 256MB and a hard disk drive of at least 10GB.

TABLE III
THE SYSTEM ATTRIBUTES FOR THE DESIGNED ENCODIA

Attribute	Constraints
Response Time	The system will display output within 3 seconds after the submission of the input
Design constraints	Encodia shall be designed with Object Oriented design approach/methodology
Operating System	Encodia will operate on windows NT/2000, windows XP and Vista with .NET framework 2.0 and above
Robustness	Encodia will work even if there is a wrong input
Easy of Use	Encodia will be self explanatory that will enable a person with basic computer knowledge to use it

As well known, a use case is a typical sequence of actions that an actor performs in order to complete a given task. An actor is simply the user of the system [3]. By building up a collection of use cases, we can describe the entire system we are planning to create, in a very clear and concise manner. Use case diagrams are UML's notation for showing relationship among a set of use cases and actors. The notation is used as an oval represent a use case with a stick figure representing an actor and a line between actor and use case representing that the actor participates in the process. The use case diagram for the Encodia software is shown in Fig. 1.

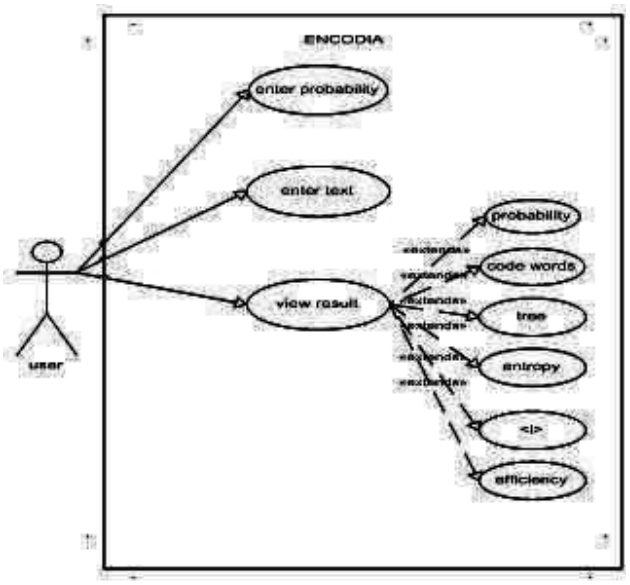

Fig. 1 Encodia Use Case Diagram

The use case descriptions follow the following nomenclature: (1) the use case name gives the unique identifier for the use case; (2) the actor(s): Models anything outside the system that need to interact with the system; (3) the description provides quick overview of the system process; (4) the assumptions tells that something has been assumed that it has occurred ; (5) the preconditions define all the conditions that must be true for the trigger to cause the initiation of the use case; (6) the post conditions explains the changes left in the system as a result of execution of a particular use case; (7) the main flow of events is a typical course of events; (8) the eexceptional flow of events shows the extra paths of the system events when something goes wrong; and (9) the cross reference implies the related use cases or system functions.

The Encodia use cases are described in Tables IV, V, and VI.

TABLE IV
ENTER TEXT USE CASE

Use case name	**Enter text**
Actor(s)	Student/lecturer/tutorial assistant/any user
Description	The user enter text/symbol for encoding
Assumptions	The entered text/symbol is in acceptable form
Preconditions	Correct text entered
Post conditions	Encoded text returned
Main flow of events	The system displays the main interface; the user select his/her desired option, enter the text, the system check correctness of the input and encode it.
Exceptional flow of events	The user cancels the process
Cross reference	Function R4.1, R4.2, R4.4, R4.6, R4.12, R4.14, R4.15

TABLE V
ENTER PROBABILITY USE CASE

Use case name	**Enter probability**
Actor(s)	Student/lecturer/tutorial assistant
Description	The user enter probability of source symbols for encoding
Assumptions	Entered probabilities are in the required form
Preconditions	Correct probability entered
Post conditions	Encoded symbols returned
Main flow of events	The system displays the main interface; the user select his/her desired option, enter the symbol probability, the system check correctness of the input and encode it.
Exceptional flow of events	The user cancels the process
Cross reference	Functions R4.1, R4.2, R4.1, R4.3, R4.6, R4.15

TABLE VI
VIEW RESULTS USE CASE

Use case name	**View result**
Actor(s)	Student/lecturer/tutorial assistant
Description	View the encoded symbols as code words
Assumptions	The symbols are encoded the required form
Preconditions	Symbols encoded
Post conditions	Encoded symbols were displayed
Main flow of events	The symbols were encoded to the required form and then displayed
Exceptional flow of events	The user cancels the process
Cross reference	R4.5, R4.7, R4.8, R4.9, R4.10, R4.11, R4.12, R4.13, R4.14

A system sequence diagram is a description of what the system does. It identifies and illustrates the system operations regarding user requests without explaining how it does [1]. For the proposed Encodia software, the sequence diagrams are shown in Figs. 2 through 4. The system is implemented as a black box.

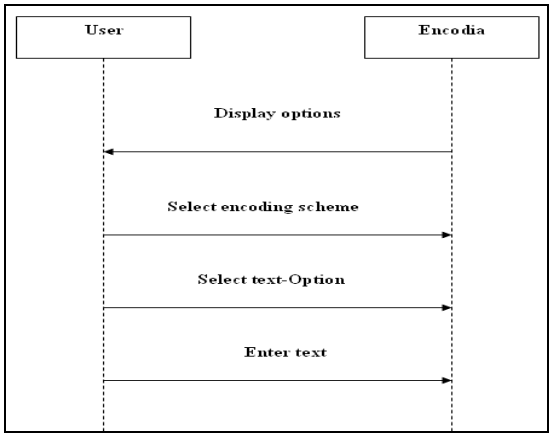

Fig. 2 System Sequence Diagram For Enter Text Use Case

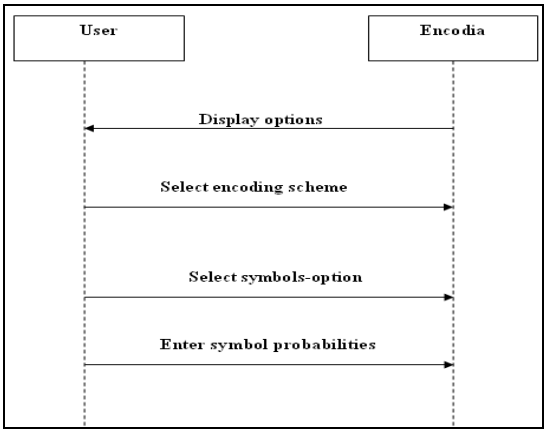

Fig. 3 System Sequence Diagram for Enter Probability Use Case

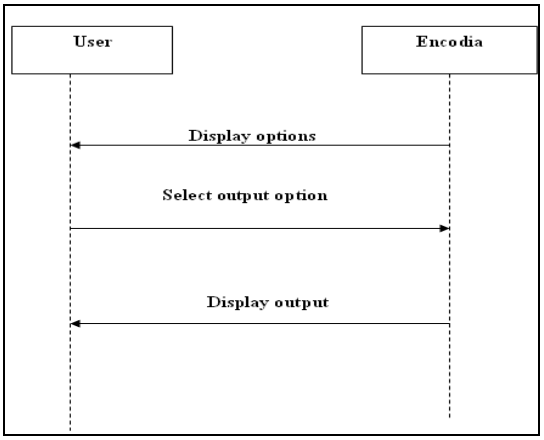

Fig. 4 System Sequence Diagram for View Result Use Case

To explain about the system class diagram, we note that a class is a representation of an object and, in many ways, it is simply a template from which objects are created. Classes form the main building blocks of an object-oriented application. Class diagrams are the mainstay of object-oriented analysis and design. UML class diagrams show the

classes of the system, their interrelationships, the operations and attributes of the classes. Class diagrams are used for a wide variety of purposes, including both conceptual modeling and detailed design modeling.[3]. Fig. 5 shows the class diagram for the Encodia software designed.

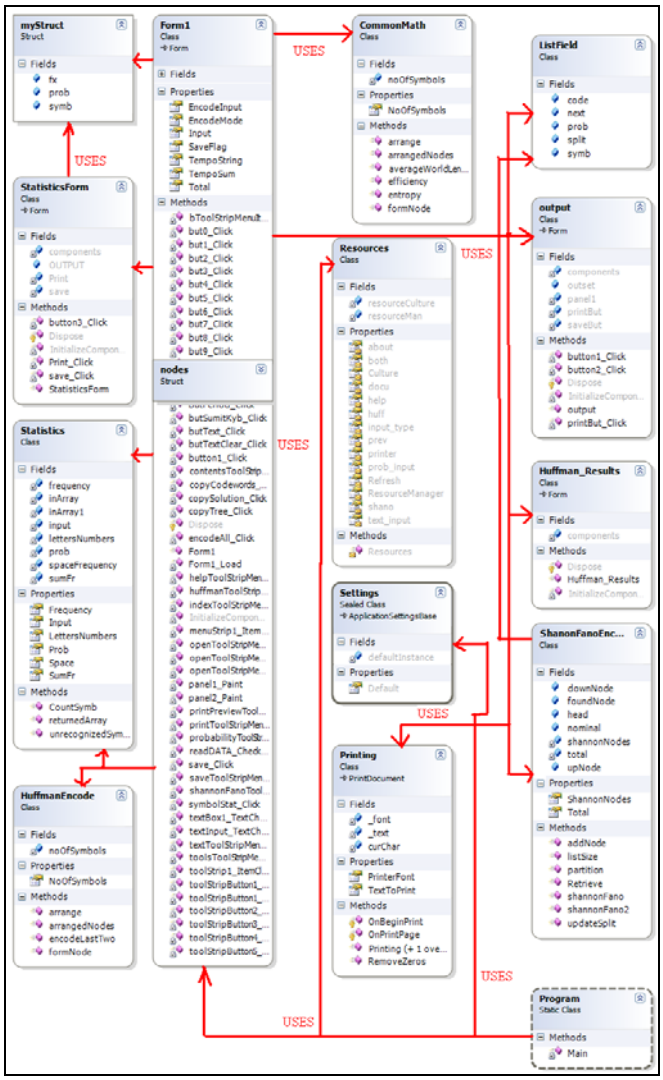

Fig.2 Program Structure (Class View)

V. System Implementation and Analysis

The flow chart of the implementation of the design is shown in Fig. 6. This flow chart stipulates how different program elements do interact with one another to complete the design goal.

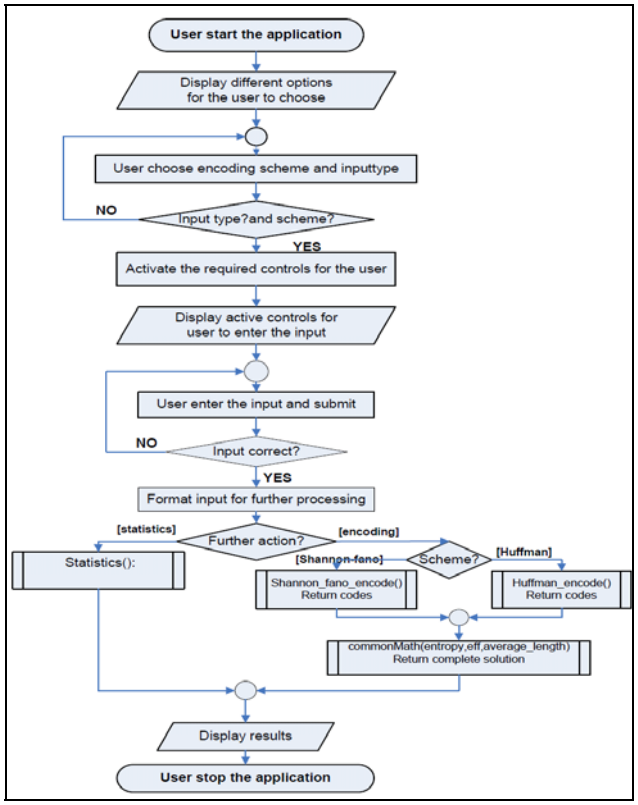

Fig. 6 Program Flow Chart

The user interface provides the means by which the user can interact with the system. The user provides two ways to do so. If the user needs to write a text, then a text area is provided, but it has to be activated by selecting the input type as text. If the user needs to enter symbol probabilities, then a keypad is provided, this is activated by selecting input type as symbol probability. Fig. 7 shows this user interface. The program has been given the name “Encodia” based on the Encoding activity it performs.

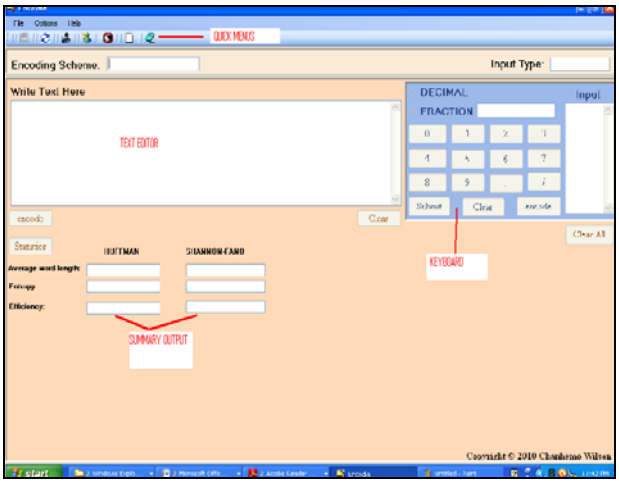

Fig. 7. Encodia User Interface

VI. System Testing and Analysis

In testing the Encodia program, two approaches were used – the module (unit) testing and the system testing. While in module testing all methods which appear in the program structure in Fig.7 were tested during development so as not to accumulate many errors at system testing level, in system testing the testing of the whole application has been done. After all functions and procedures have been combined to make a single application, Encodia application was tested to see if it provides the required functionality. In testing Encodia, three approaches were employed. These are:

(a) The probabilities of symbols from a known source were entered to the program and their codewords computed by the software, the results compared with manual computation. Example in Fig. 8 shows how it was calculated. Given symbol probabilities as $S_1 = 0.4$, $S_2 = 0.2$, $S_3 = 0.2$, $S_4 = 0.1$, and $S_5 = 0.1$, the manual computation for Huffman algorithm gives the results as in Table VII.

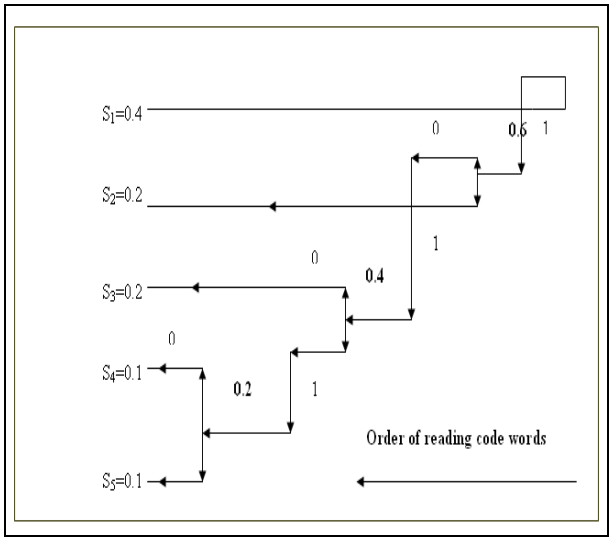

Fig. 8 Huffman Encoding Manual process

TABLE VII
HUFFMAN ENCODING MANUAL RESULTS

Symbol	Code word
S_1	1
S_2	01
S_3	000
S_4	0010
S_5	0010

(b) The software approach where the implemented Encodia was used to calculate the codewords as obtained by Huffman algorithm after inputting the symbols' probabilities. Figs. 9 through 11 illustrate the presented approach. We can also check the values for the source entropy, average word length and efficiency.

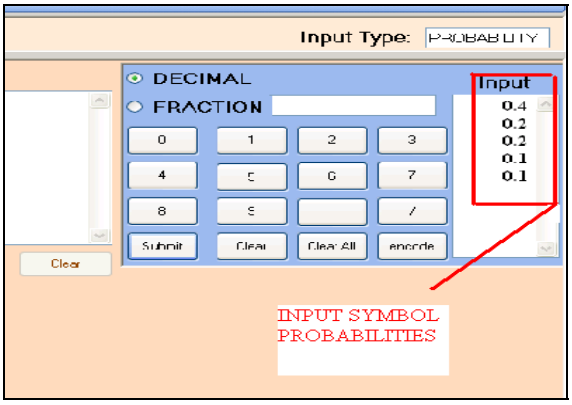

Fig. 9 Huffman Encoding Using Encodia Software

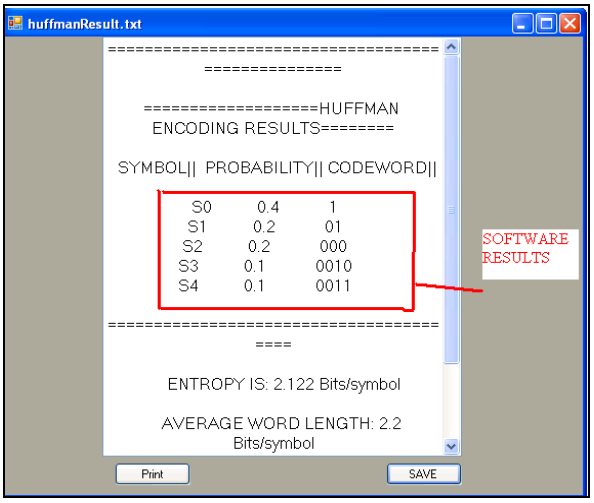

Fig. 10 Output from Encodia Software

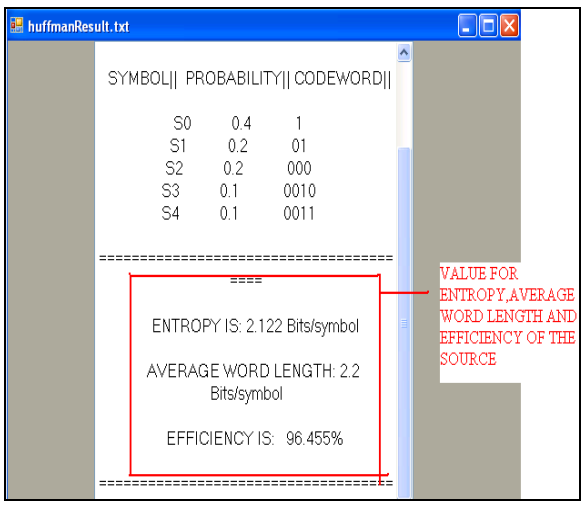

Fig. 11 Summary Results

(c) The software approach where the implemented Encodia was used to calculate the codewords as obtained by Shannon-Fano algorithm after inputting the symbols' probabilities. Figs. 12 and 13 illustrate the presented approach. We can also

check the values for the source entropy, average word length and efficiency.

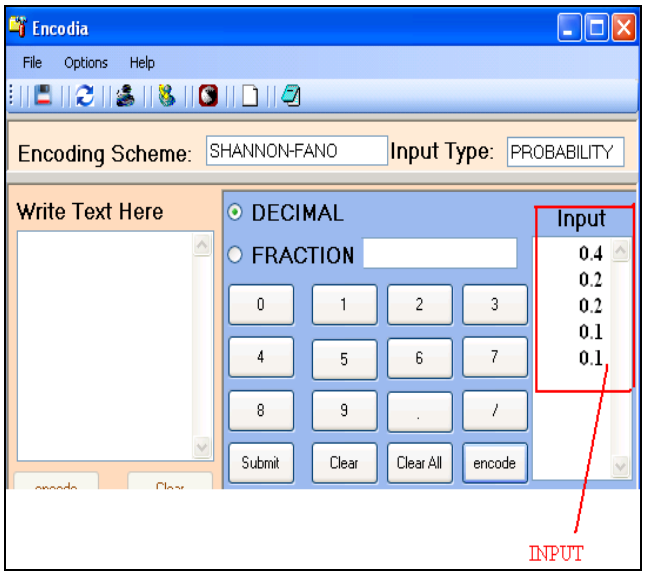

Fig. 12 Shannon-Fano Encoding using Encodia

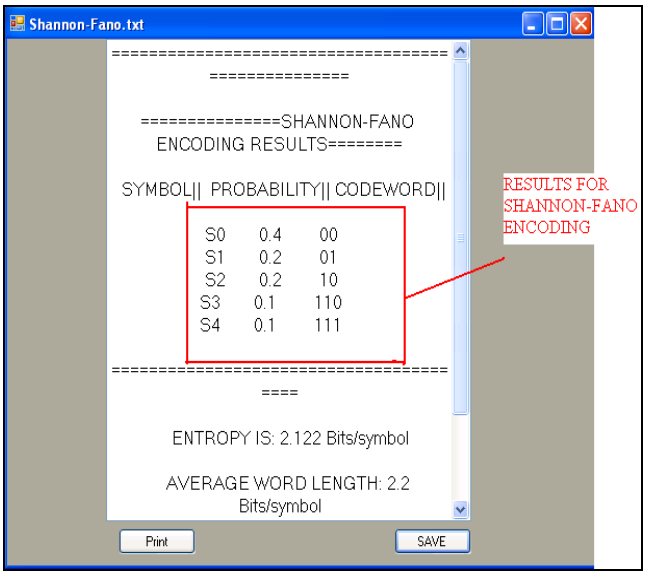

Fig. 13 Shannon-Fano Encoding Results.

VII. System Deviation

Encoding using Huffman algorithm gives the exact codewords as the one that can be obtained manually. Nevertheless, recalling how Shannon-Fano algorithm works [4], Fig. 14 shows that the grouping of symbols by the system always checks that the sum of the first group probabilities is exactly greater or equal to half, hence for the example under discussion, symbol with probability 0.2 will be placed at the lower group instead of the upper group as one will do by manually. This action will cause some symbols to have different codewords as compared to one doing by hand.

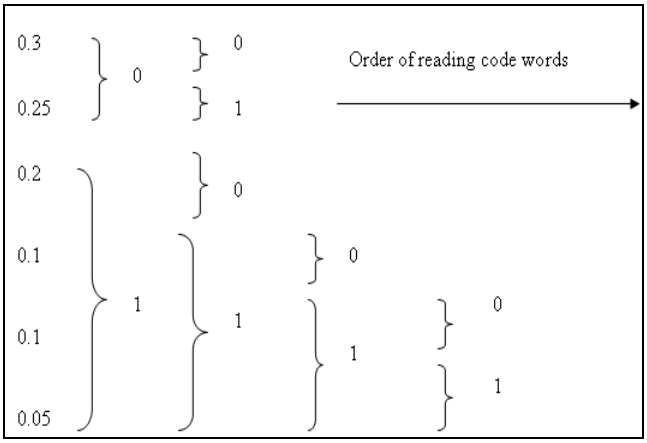

Fig. 14 Shannon-Fano Encoding Process

So, when implementing this algorithm, the user will notice some deviations especially for intermediate symbols where there is possibility of placing them in either of the two groups. The user, so, should not worry about this deviation because it's the matter of judgment taken by the system which may sometime not necessarily be the same as his/her judgment.

VIII. Compatibility Testing

This testing is conducted on the application to evaluate the application's compatibility with the computing environment. Encodia has been tested and works under Windows XP, Windows Vista and Windows 7 Operating Systems, also was tested in IBM compatible PCs and found to work correctly.

IX. Comparison to Some Related Works

In comparison with some related works, the Encodia is found to be far better application to demonstrate encoding as these few tips shows here

i. Encodia application is written using object oriented language which makes it simple to maintain, modify the code and easy to understand the source code
ii. Encodia application have Graphical User Interface which makes it easy to learn and use it compared to many projects which are command based with no read me files for new users
iii. Encodia application is not limited to number of symbols to encode as compared to other applications or size of a given text.
iv. Encodia application have power to use multiple encoding at once hence it help users to decide on the best encoding scheme based on a particular text/ symbol probabilities
v. It is not limited to known symbols probabilities only, even a piece of text can be encoded after stepwise analysis to obtain each symbol's probability of occurrence
vi. Encodia application provides the user with read me files which help even the new user to be conversant with it in few minutes

The Encodia is a Multi purpose application; it can be used in class as a learning tool or a research tool.

X. Conclusions and Future Direction

Encodia software has been developed, tested and found to meet the intended requirements. It is my expectation that this application will help students and teaching staff in their daily doing of Information Theory. My intention is to modify this tool so that it can also be useful in research activities on source coding more intensely. This work has started and I hope it will find its final destination intended.

Acknowledgment

This work has been partly supported by the College of Engineering of the University of Dar es Salaam, Tanzania and the South African National Research Foundation through the University of Johannesburg's Research and Innovation Division (RID).

References

[1] Liu Z, Object Oriented Software Development Using UML, UNU-IIST, Macau, 2004
[2] Robert J.M, A Theory of Information and Encoding, Addison-Wesley Publishing Company, Reading Mass, 1977
[3] Schach S.R, Classical and Object-Oriented Software Engineering with UML and Java, Inc.McGram-Hill Companies, Boston, 1999
[4] Van de Lubba and Gee S, Information Theory, Cambridge University Press, Cambridge, 1997
[5] Rai, Gandalf, Huffman encoding in C (Minimum variable encoding), 30th November, 2006
[6] Rajiv A Iyer, Huffman encoding Implementation in C, TE Comps, SIES GST, Nerul, 2006
[7] Michael Vonshay Cooperwood, Sr, Analysis and performance comparison of adaptive differential pulse code modulation data compression systems, Thesis report, Naval Postgraduate School Monterey, California, March, 1996
[8] Davis, Shannon-Fano encoding Implementation in C, Engineering student in India's Blog,, 2006
[9] Robinson S, et all, Professional C#, Wiley Publishing Inc, Indianapolis-India, 3rd Edition, 2004.
[10] Sells C. and Weinhardth M, Windows Forms 2.0 Programming, Addison Wesley Professional, USA, 2nd Edition, 2006.
[11] eBook, Coding Theory: Algorithms, Architectures and Applications Wiley-Interscience (December 4, 2007) ISBN:0470028610

Wilson Chanhemo graduated his Bachelors degree in Telecommunications Engineering from the College of Engineering and Technology of the University of Dar es Salaam, Tanzania, in 2010. His research interest seems to be in the fields of Information Theory and Coding, Telecommunications Systems, Computer Programming and Software Development for practical Telecommunications systems.

Henry R Mgombelo holds a Ph.D. degree (1983) from the University of Bradford, U.K. having specialized in mobile radio communication circuits. Prior to that he graduated from the Leningrad (now St. Petersburg) Electrotechnical Institute of Communications, Russia in 1977 with an M. Sc. (Eng) degree in Radio Communications and Broadcasting having specialized in Television Engineering. During his technical carrier Prof. Mgombelo has been teaching electronics and telecommunication subjects at the University of Dar es Salaam, Tanzania for over 30 years. He has also served in many regional and national boards and institutions including Commissioner of Tanzania Broadcasting Commission, Commissioner of Tanzania Communications Commission, Director and Vice Chairman Tanzania Electric Supply Company (TANESCO) Limited, Chairman Tanzania Cables Limited, and Director Tanzania Broadcasting Services. From 2000 to 2005, Prof. Mgombelo took leave from his teaching duties and became M.P and was elected Chairman of the Parliamentary Economic Infrastructure Committee which, among other things, is responsible for overseeing ICT matters. Prof. Mgombelo is a Registered Engineer in Tanzania and Fellow of the Institution of Engineers Tanzania (IET).

Omar F Hamad (SM'1997-M'2009) graduated PhD from Multimedia Data Communications Lab in the School of Electronics and Computer Engineering at Chonnam National University – Korea – in February, 2008. He has, since July 2002, been a Telecommunications Engineering Lecturer at the University of Dar es Salaam. He is now an IEEE Member and he has been IEEE Student/Graduate Member since 1997. He is a member of NEPAD Council and a Technical Committee of ICTe Africa Conferences and Workshops and other international conferences. His research interest is in the fields of Bandwidth Calculus in Overlay Multicast, Multimedia Systems, RDMA, FTTH Technologies, Multimedia Delivery over PLC Networks, and Telecommunications Systems. He is now a Post Doctoral Fellow at the University of Johannesburg, South Africa.

Tshilidzi Marwala is currently a Fellow of the CSIR and the Executive Dean of the Faculty of Engineering and the Built Environment at the University of Johannesburg. He holds a Bachelor of Mechanical Engineering with a Magna Cum Laude from Case Western Reserve University, a Master of Engineering from the University of Pretoria, a PhD in Computational Intelligence from University of Cambridge, and was a post-doctoral research associate at the Imperial College of Science Technology and Medicine of the University of London. He has successfully completed a Program for Leadership Development at Harvard University. In year 2006-2007, he was a visiting fellow at Harvard University and in year 2007-2008, he was a visiting fellow at the University of Cambridge. Prof. Marwala has received over 41 awards; has published over 170 articles in refereed international journals, conference proceedings and book chapters and has successfully supervised more than 30 postgraduate students at masters and PhD levels and has collaborated with more than 44 national as well as international researchers. His research interests include the application of computational intelligence to engineering, computer science, finance, social science and medicine. His work has been featured in magazines such as Time Magazine, New Scientist and ACM Tech News.