International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 8– No.4, February 2015 – www.ijais.org

Unmatched Qualities of Software Developers and Impacts of their Thinking on Coding

Adenugba, D.A.
Department of Physics
The Federal University of Technology, Akure
P.M.B. 704, Akure, Ondo State, Nigeria

ABSTRACT

This paper discusses uncommon qualities of Software Developers and impacts of their thinking on coding. Applications of Software Developers' thinking yield a novel code component (CC) for optical models of Snell's law, refractive index of a prism and Brewster's law; and enhanced server by the inclusion of Pontes et al. rainfall height models in an existing class library. Software Developers(SDs) are generous giver of comfort, speed and accuracy without racist discrimination through CCs and software applications. The quality of any application is a reflection of the tastes, thinking and qualities of SDs. SDs think for others, a difficult obligation to achieve, yet a mandatory task to perform to prevent package crash. The optical server is exposed in a new client application while the rain height codes are tested via an existing application, now known as RainHeightSoft 1.0. The accurate results computed corroborate existing facts regarding the optical models. The Pontes et al. rain height models are applied to twenty sites and findings depict that Pontes rainy model (PR) varies between 3.25-4.90; and the non-rainy model (PNR) is between 3.64-4.89. Uniformly, PR is greater than PNR for Nigeria and Brazil sites, but the opposite is the case for South-Africa. Rainfall height does not increase with increasing Latitude, rather the smaller the latitude, the higher the rain height. These useful information could be used for communication system design. Besides, dynamically, all the models' workings are generated that could be utilized real-time for teaching-learning. SDs will find our CCs useful in their work. Physicists could seamlessly employ the resulting package in teaching optical and rain height models.

General Terms

Latitude, Optics, Rain height, Refractive index, Software Developer

Keywords

Pontes et al. rainfall height models, Brewster's law, RainHeightSoft 1.0, Snell's law

1. INTRODUCTION

Thinking is an imperative part of life, especially during coding and software development. Thinking is a function of the heart, not of the stomach (Matthew 12:34-37). Generally, each person is a product of his thinking: "For as he thinks in his heart, so is he…" Thus, "Keep your heart with all diligence, for out of it springs the issues of life" (Proverbs 23:7; 4:23).

Some tasks, either simple or complex, are performed swiftly or slowly repeatedly; and some occasionally and sluggishly or quickly. Some tasks demand absolute concentration because life is at stake if the slightest error or mistake occurs doing it. Tasks widely vary in nature and in timing. Whatever the nature of the task to perform, it is expected to be carried out in shortest possible time and with maximum comfort and accuracy. When a chore keeps reoccurring on daily basis, it soon becomes boring and uninteresting to an average soul. Efficiency and health suffer greatly from it under this repetitive task condition(s).

Indeed, monotonous tasks are generally unpleasant to an average folks. The indisputable solution to this situation is a flexible and accurately working software package, which gladly handles repetitive chores with absolute accuracy swiftly. Software applications leverage individual, group of people and nations to achieve uncommon feats. Indeed, national capacity building and self-reliance are anchored on accurately functioning software packages [1].

A Software Developer (SD) uses codes to solve human problems [1-6]. Andy (2014) [7] captures one of the fundamental goals of code component when he said, "With code available for our use that has already been written, tested and debugged…, you're almost always better off looking for an existing solution than trying to code one yourself…You'll spend far less time finding an existing library and learning to use it rather than trying to roll your own code…" The implication of this vital statement is that Software Developers (SDs) should not re-invest the wheel, thereby saving invaluable time and efforts. The use of existing flexible and accurately working class library or libraries saves considerable time and guarantees reliability [1-2].

In this paper, we discuss the sterling qualities of SDs and how these sway their decisions at every stage of designing, writing of codes, developing of custom methods and non-crashable applications. Specifically, we apply these unique qualities and thinking of SDs to code components development for the estimation of Snell's law, refractive Index of prism and Brewster's law. A client application, OpticsSoft will be developed using Microsoft Visual Studio, 2013 to test the optical class library, dvOpticsCls. In addition, codes will be written for Pontes et al. rainfall height models, which will be exposed through an existing software application, HeightSoft. Results will be computed for twenty sites and the Pontes et al. rainy model(PR) will be compared with Pontes et al. non-rainy model(PNR). In the process of these activities, SDs thinking and nature will be exposed.

2. SOME QUALITIES AND THINKING OF SOFTWARE DEVELOPERS

SDs have unique ability to give unique names to data, properties, functions, procedures etc such that they will be reminded about what the object represents, and function(s) it performs. Good names given to objects do not only assist code readability, but enhances code maintainability and comprehension. A SD who fails to meaningfully name objects in his/her codes creates confusion, chaos and problems later in maintaining the life-cycle of the code or project. Good object-naming is a clear manifestation, to a reasonable degree, of good programming skill and practice. Professional SDs are distinguished from quacks through good naming styles employ in his/her code. “A developer”, as John (2012)[5] aptly observed, “who lacks the ability to give good names to concepts and data in their code is like a mute translator.”

SDs are enterprising people, full of ideas. Software Developers possess the pipe and dictate the fine desired tone in virtually every spheres of life. Developers are revenue generators. Software packages are sure sources of revenue for organizations and the nation. In 2012, India expected software exports revenue generation is $70 billion (₦10.5 trillion) which is far above most African nations’ annual budget [8]. How much US earns every year on software could be best measured against the 12th position of Bill Gates on the list of the 25 richest people who ever lived on earth [9]. That Bill Gates is the richest living American with net worth of $136 Billion is an eloquent testimony to the power of software to earn astonishing revenue for any individual, besides companies and nations [9]. Although software professionals are prestigious and well-paid compared to other professions in India, only a few women work as SDs, the ratio of men to women being 76:24 [10].

Statistical figures provided in recent times show that software industry is going to be leading for a long time to come in job creation and employment. The predicted job growth for computing professions from 2010-2020 shows that Database Administrators is 1% ahead of Software Developers with 30%; this is followed by Network Systems Administrator with 28%; while both Web Developers and Computer Systems Analysts have 22% each; the least of 19% goes to the Information Research Scientists [11,12]. What this translates to is that there will be a steady job growth in the computing world with median pay steadily rising. To stay with the progressives with secured job, you need to be in the computing professions, especially software development, thinking for others and developing application to earn living and advance researches.

To catch up with the Superpowers, it has been correctly observed that developing nations should fully plunge themselves into software development and copiously invest in it [1].

Patience, yes, perseverance, is an attribute every serious-minded SDs should possess in order to succeed in their career. When codes stubbornly decline to do what is required, patience is needed; step-by-step debugging of the codes is a must; ability to swiftly recall previous experience(s) is quite helpful and useful to isolate the culprit areas needing attention.

SDs think rationally and uniquely. They quest for alternatives to achieve the most efficient algorithm to solve problems. Therefore, SDs are not parochial; thus they explore various avenues to achieve a task swiftly, accurately and efficiently [4-6].

Software applications are capable to redefine things and redirect folks outlook and thinking. Indeed, not only Science could software redefine, but virtually all things human mind could conceive.

SDs are lenient person who do not terminate a malfunctioning program without providing adequate information on what went amiss. When necessary, they ignore invalid inputs and picks another one that is valid for use as done in the loop of the codes in appendix A for Snell’s law computation. Please, see the attached comments to the codes for more explanation.

SDs think for others; thus very proactive. Thinking for others is a difficult task. Yet, SDs have to think ahead for users on their input(s) and output(s) needs, behaviours and attitudes. As for data input, SDs trap all likely errors user is capable of, and likely to, commit. They extensively perform data input checks often before releasing any package into the market. These checks bloated the size of the codes, and they are modularized, so as to allow other SDs; and methods and functions to make use of them. For instance, dvCalOpticsModels method, that is called by dvCompSnellLaw function of appendix A to calculate a single result of Snell’s law, accepts an enumeration class of five members, which indicates what to calculate. This method checks the inputs; if invalid, it returns NR (for No Result) else it computes result and returns it. See row 2 of Table 1 when this situation occurs due to invalid incident angle supply (dv). When either the incident angle, i or refracted angle, r is not supplied, NN (for Not Numeric) is returned and stored in the output columns as seen in Table 2(2nd row).

SDs always expect the unexpected in the use of their applications, thus they trap all humanly conceivable errors and mistakes that could be made and rendered the applications useless during use. They wrap the codes around error trapping statements such as Try…Catch…End Try and On Error ….GoTo statements. This is to capture the unexpected error(s) and report back to the calling program as can be seen in the codes in Appendix A.

Still on data checking, SDs stoutly believe in quality of data, hence they filter input data every time. Data manipulation is as significant as data acquisition and transmission. Filtering of data is one of such data manipulation that improves or enhances the data for use. Data check is a must for all SDs who want to succeed.

Availability of code components often motive SDs to develop flexible and accurate software applications [1-2]. Because of available numerous functioning code components, many are software applications developed to meet the yearning of people. SDs design, develop and make available code components for the use of others. Thus, they are quite industrious, observant and considerate.

SDs seek to stay abreast of the latest news, and swiftly learn the latest techniques and technologies that will enhance their productivity and skills. Thus, a membership of professional society is a must, such as Association of Computing Machinery (ACM) and/or IEEE(and other ones around the globe). SDs read good and relevant books. “Most highly skilled developers”, according to John (2009)[4], “will have a library of books” that he/she reads and references frequently. Three basic things are well-known to all great SDs. These germane facts are: they always know that there is someone to learn from, something *more* to learn, and they hunger to do things the right way[4].

SDs experiment with variations and come up with patterns or styles they enjoy most and that increase their effectiveness [13]. In other words, applications mirror their developers’ thinking.

SDs are extremely orderly and cautious to assign values to arguments in order to avoid getting unexpected results. A variable that is declared at form's or server's level and is passed as argument before it is assigned a value has zero value (or for object data type, it is internally assigned Nothing) and generate wrong results or throw an exception; if the expected value(s) is not assigned to it before use.

3. CODING

SDs cherish an item that reoccurs, since a separate function or sub-procedure can be written for it and calls in several places within the project and/or in another projects. For instance, the angle in degrees gotten from the user via an overloaded method, dvSetInputs, has to be converted to radians before passing it to the functions that calculate prism refractive index and Brewster' law. The dvSetInputs property accepts one of five inputs, depending on users' need; then a new method was called to convert from degrees to radians; the old method being inadequate to generate workings. Since the previous method has no dynamically working generator, three additional methods were written to do the conversion and generate workings for each computation. The inclusion of an enumeration class to indicate if input should be zero or not is an added benefit to filter data before use and report back if error occurs. The old method is intact, but Overloads keyword preceded the function keyword, so that the methods are polymorphized. Needless to mention that these new method will find extensive use in estimating trigonometrically functions and produce the workings that can be used for teaching-learning real-time.

The eight methods written for sine and tangent estimation have the same reference outputs result (Ans) and workings (Wkgs), but varying inputs. For both, we have this code pattern, which makes it easy to call and maintain:

dvCalSinX or dvCalTanX

i Input1, Ans, Wkgs

ii Input1, Input2, Ans, Wkgs

iii Input1, Input2, Constant, Ans, Wkgs

iv Input1,Constant, Ans, Wkgs

The first (i) will solve for any Sin(X) or Tan(X). (ii) will solve for any Sin(X+Y) or Tan(X+Y); (iii) will solve for any Sin((X+Y)/C) or Tan((X+Y)/C) similar to the numerator of the prism refractive index model and (iv) will solve for any Sin(X/C) or Tan(X/C) identical to the denominator of the prism refractive index model. X and Y are the object inputs; and being object, they will conveniently accept integer and floating data types. These methods are handy for use where sin and tan are found in any equation. Not only will the computed result be returned, the workings will also be outputted.

SDs are not presumptuous but factual. They do not assume, for instance, that what solves for X/C will do for C/X until they have tested carefully for all the involving parameters before arriving at informed decision(s). If the Mathematical operator is * instead of /, then a single function will produce the same result, else different treatments will have to be accorded to the equations.

SDs are not dense, follow-follow type, but fast thinking folks who make code clarity their watch word. As time progresses with experience, SDs develop coding style(s) that give them uncommon gratification and comfort. The quality of Application developed reveals who a SD is; his taste for well-blended colours, quality and thoroughness. Therefore, "Application-building process," according to Tim (2008)[13], "is about much more than syntax, statements and logic." It exposes who the Developer really is in terms of quality and capability to use existing and self-developed tools to produce first class application that pragmatically addresses human needs. Discipline, planning and ethics are three traits which had been identified to provide a strong basis for programming life; and deficiency in any, according to Tim (2008)[13], results in poor and deficient application and code. These traits make programming work so much easier, comfortable and enjoyable; and positively impact the other areas of useful earth life [13]. SDs are agog and delighted in assisting "People become more productive through specialized or general software" [13].

SDs are not angels; they do make mistakes. But when they do, they quickly identify and debug them. Prudently they bookmark bugs, make list of silent bugs that compiler fails to adequately reveal. One of such stubborn mistakes is in reference to Microsoft DataGridView Control (DGVC):

InputToUse = DGV.Item (Column, Row)

This line of code is not highlighted during coding as if it is okay. Instead of the required value, identified by the zero-based column and row, to be assigned to InputToUse object variable, an alien string is obtained. This is simply because the *Value Property* is inadvertently omitted. The good counsel of John (2013)[6] is appropriate here: "Don't be paranoid about making mistakes and failing---fear will paralyze and destroy your progress, much more than mistakes will, but, instead be cautious, careful and deliberate in your actions. If you make a mistake, learn from it". When using DGVC, therefore, never fail to zero-based the column and row; remember that column precedes row and without the *Value* property (*InputToUse = DGV.Item (Column, Row).Value)*, no valid input is fetched from the DGVC.

Another thing that characterized the thinking of SDs is to always close opened files when done with them. All objects such as pen, brush etc in Graphics Device Interface Plus (GDI+), as well as database objects like DataAdapter and Connection objects should be closed and disposed of to free memory for other uses. Therefore, all SDs always remember certain things during coding and before packaging. Indeed, at design time a list of useful coding parameters should be jotted down. This list will vary from one project to another. However, there are certain things that are common to average projects such as opening and closing files, which SDs never toy with.

All rational SDs never waste functioning codes. These codes may not have instant use, yet they keep them for later use or for neighbour's (other developer's) requests and needs. Hence, SDs have repository of codes and working functionalities that can be summoned at any given time for use. Because they are not selfish, SDs generously assist others with their well-crafted working tools. Also, being not a father-Christmas, SDs do not just throw about codes. By Inspiration Matthew wrote, "Therefore, "Ask, and it will be given to you; seek, and you will find; knock, and it will be opened to you" (Matthew 7:7). Similarly, blessed are those who ask for server methods for they shall receive working class library of them. Blessed are those who seek for code-help for they shall receive myriad of them. Blessed are those who knock the door of functionalities for they shall receive uniquely working functionalities written in well-known computer language(s).

4. DEVELOPING THE SERVER AND PACKAGE

A class library, dvOpticsCls was developed for three optical models: Snell's law, refractive index of a prism and Brewster's law. Apart from dvSetInputs method mentioned earlier, the overloaded dvSetDGVTitle method inserts title in DGVC. It allows user to vary DGVC titles as seen in Tables 1-6. The dvChkDGV function checks the DGVC to be valid for input columns. If the column(s) in the DGVC is less than the required for data inputs, error message is given and the program terminated. For single result computation, a single function each is adequate to handle the models; also, three functions for multiple result generation, but only two methods. Where applicable, the functions call a method to perform degree to radian conversion. The dvCalOpticsModels method for single result accepts an enumeration class of five members and switches to the appropriate function, which computes results and returns it, together with the workings. The second overload of dvCalOpticsModels method computes multiple results, base on Enumeration Type specifies and returns both input and result (s) in a reference DGVC.

A client application was developed with Microsoft Visual Studio, 2013 to expose the functionalities in dvOpticsCls library. When the Equations main menu is clicked, six submenus are displayed. Each of them will generate the models and combinations of the models.

The Computations main menu has seven submenus under it. Each of the submenus has two submenus of Single Result and Multiple Interactive Data Entry Result. The results in Tables 1-6 reflect the SDs' thinking. The accurate results in Tables 1, 3 and 5 are enough to show the optical models results, but in order to provide more information that could aid learning and teaching real time, Tables 2, 4 and 6 are provided, which display the various parts of the equations. Note that the last columns of all the tables are identical. By showing the various parts, we have saved a lot of time and efforts during learning and teaching. What is displayed depends on the user's need(s). Here, we resolve to provide swift teaching-learning situation, so it is appropriate to show the parts. Also, all the times any of the optical model is computed, the equation is shown to assist learning and teaching through a property summoned from the dvOpticsCls library.

Table 1: Snell's Law, n No Parts

i	r	Snell Law, n
dv	30	NR
30	52	0.63
23	40	0.61
90	12	4.81
77	83	0.98

Table 2: Snell's Law with Parts

i	r	Sini	Sinr	Snell Law, n
	30	NN	NN	NN
30	52	0.5000000	0.7880108	0.63
23	40	0.3907311	0.6427876	0.61
90	12	1.0000000	0.2079117	4.81
77	83	0.9743701	0.9925462	0.98

Table 3: Prism Refractive Index, n No Parts

A	D	n
70	55.00	1.5464562
43	8.45	1.1843131
33	42.00	2.1434103
27	60.00	2.9486753
80	39.23	1.3420391

Table 4: Prism Refractive Index, n with Parts

A	D	Sin(A+D/2)	Sin(A/2)	n
70	55.00	0.887010833	0.573576436	1.5464562
43	8.45	0.434052212	0.366501227	1.1843131
33	42.00	0.608761429	0.284015345	2.1434103
27	60.00	0.688354576	0.233445364	2.9486753
80	39.23	0.862646119	0.642787610	1.3420391

Table 5: Brewster's Law, n No Part

i	n = tani
40	0.8390996
70	2.7474774
84	9.5143645
63	1.9626105
12	0.2125566

Table 6: Brewster's Law, n with Part

i	Radian_i	n = tani
40	0.6981317	0.8390996
70	1.2217305	2.7474774
84	1.4660766	9.5143645
63	1.0995574	1.9626105
12	0.2094395	0.2125566

5. RAINFALL HEIGHT

HeightSoft is a software package for computing Ajayi-Barbaliscia, International Union of Telecommunication Radio group (ITUR) and Sarkar rain height models. The need to include more rain height models in this package has been pointed out [14]. We now include Pontes et al. rain height models and rename the application as RainHeightSoft 1.0. Rain height, on which slant path rain attenuation prediction depends, is regarded as the 0^oC isotherm height; beyond which there is no rainfall, hence no rain attenuation due to the absence of rainfall [15]. Data from three countries are used in this work from ten Nigeria sites; five each from Brazil and South-Africa.

Six menus for single and multiple results are added to the package which screenshot is not shown for space, but see[14]. The models could be displayed singly and in combinations. For instance, the AB_Pontes Models submenu produce this result, which when compared with the models' sources [15-17] are faithfully reproduced.

Ajayi-Barbaliscia Northern Hemisphere

Rain Height,hfr = 4.6 - (0.084 * (Lat - 26.0))

Ajayi-Barbaliscia Southern Hemisphere

Rain Height,hfr = 4.6 - (0.1 * (Lat - 26.0))

Computing Pontes et al. Rainy Model; 0 <= Latitude < 23

Rain Height,hfr = 4.9 - (0.004 * Lat)

Computing Pontes et al. Rainy Model; Latitude >= 23

Rain Height,hfr = 7.3 - (0.115 * Lat)

Computing Pontes et al. Non-Rainy Model;

0 <= Latitude < 20

Rain Height,hfr = 4.9- (0.009 * Lat)

Computing Pontes et al. Non-Rainy Model;

Latitude >= 20

Rain Height,hfr = 6.0 - (0.067 * Lat)

Also, for Sarkar Models SM_SS_SW, we have:

Sarkar Monsoon Model (SM)

Rain Height,hfr = 4.063 + (0.146 * Lat) - (0.0935 * Lat * Lat) + (0.000334 * Lat * Lat * Lat)

Sarkar Summer Model (SS)

Rain Height,hfr = 3.733 + (0.309 * Lat) - (0.02 * Lat * Lat) + (0.000566 * Lat * Lat * Lat)

Sarkar Winter Model (SW)

Rain Height,hfr = 4.579 + (0.077 * Lat) - (0.0027 * Lat * Lat) + (0.000098 * Lat * Lat * Lat)

Where: Lat = Latitude(deg.)

Other combinations are expected for the rest models. The dynamically generated equations could be utilized real-time for learning and teaching.

When the single result Pontes et al. rainfall height Rainy model menu is clicked, a customized user interface(UI) is displayed for latitude value entry, so also for Non-Rainy model. For Ijebu-Igbo site in Nigeria, the results for both models are:

Ijebu-Igbo in Nigeria

Computing Pontes et al. Rainy Model 0 <= Latitude < 23

Rain Height,hfr = 4.9 - (0.004 * Lat)

Lat=6.59

Rain Height,hfr = 4.9 - (0.004 * 6.59)

Rain Height,hfr = 4.874

Computing Pontes et al. Non-Rainy Model

0 <= Latitude < 20

Rain Height,hfr = 4.9- (0.009 * Lat)

Lat=6.59

Rain Height,hfr = 4.9- (0.009 * 6.59)

Rain Height,hfr = 4.841

For multiple interactive data entry results, the same UI is displayed, but the customized control visible property is true instead of false as it was for single result. Also, a Compute main menu is enabled for calculation to take place. Up to two thousand interactive data entry are permitted. After entering latitude values for 19 sites and the Compute menu clicked, the results in Table 7 are generated for both PR and PNR for nine Nigeria, five both for Brazil and South-Africa sites. The station names are in column1, while the input latitude (Lat) occupies column 2; columns 3 and 4 are for the results computed.

Table 7: Pontes et al. Rainfall Height Models Results for Nigeria, Brazil and South-Africa

Nigeria Sites			
Station	**Lat**	**PR**	**PNR**
Benin	6.19	4.875	4.844
Lokoja	7.47	4.870	4.833
Minna	9.37	4.863	4.816
Jos	9.52	4.862	4.814
Toro	10.03	4.860	4.810
Bauchi	10.17	4.859	4.808
Zaria	11.05	4.856	4.801
Potiskum	11.42	4.854	4.797
Maiduguri	11.51	4.854	4.796
BRAZIL SITES	**Lat**	**PR**	**PNR**
Belem	1.23	4.895	4.889
Manaus	3.90	4.884	4.865
Fernando Noronh	3.51	4.886	4.868
Cachimbo	9.22	4.863	4.817
Vilhena	12.44	4.850	4.788
SOUTH AFRICA SITES	**Lat**	**PR**	**PNR**
Eastern CapeBhisho	35.21	3.251	3.641
Port Elizabeth	33.57	3.439	3.751
Umthatha	31.50	3.678	3.890
Port Alfred	33.50	3.448	3.756
FreeState_Bethlehem	28.23	4.054	4.109

With single and multiple results generation along with the workings, users are empowered with a tool for research and education.

6. PERFORMANCE

The thinking of SDs rob off on their applications; their works expose their tastes and qualities for such thing as colour and images; menu and controls arrangement. What are presented are the choices of the SDs to a large extent as could be seen in this work. However, the clients they develop for also dictate what they want, which the SDs should comply with. Codes are written, gleaned and tied up to form a unit whole that solves human problems. SDs are very conscious of speed wherever speed matters. Application development implies a thoughtful, dedicated and vigilant craftsmanship. Software returns more revenue than selling pepper in the market. It gives the peak of satisfaction as one sees his/her software packages running accurately. Also, any nation economy can securely depend on software application development.

It was found that our exposed functionalities in dvOpticsCls class library produced the same results as in previous works with additional flexibility of varying the decimal points via a property when compared with other previous works [18-25]. The property could be set to suit users' desire to any level of decimal points, including scientific notation. More so, the resulting client application used in testing the optical class library is very flexible to use, but not yet in its final form.

The Pontes et al. rainfall height models have two components---Rainy(PR) and Non-rainy(PNR); the PR varies between 4.85-4.87 for Nigeria, 4.85-4.90 for Brazil and 3.25-4.05 for South-Africa. But for PNR, we obtained respectively for Nigeria, Brazil and South-Africa, 4.80-4.83;4.79-4.89 and 3.64-4.11. PR is greater than PNR for all the sites, except South-Africa where the reverse is the case. Since the results here are consistently above 4km for Nigeria and Brazil, the effective rain height cannot be taken as approximately practically constant at 4km at tropical and equatorial zones [26]. Unlike Nigeria and Brazil, PNR will account for South-Africa better than PR. Up to the rain height computed, vertical rainfall structure is assumed uniform[15-16, 27], but this has been found not to be totally correct in practice [28]. To prevent unnecessary computations and results, we insert the appropriate conditional statements in our codes to check model limits.

When the latitude is small as in Belem of Brazil and Benin in Nigeria, both PR and PNR are higher than when the latitude is higher (see Vilhena and Maiduguri sites in column 1, Table 7). The higher the latitude, the smaller the rain height as depicted for South-Africa sites. More so, during wet period in Nigeria and Brazil, rain height is more than dry season. These information will aid in modeling earth-satellite slant path rainfall attenuation, which depends on rainfall height.

Instead of using DGVC, we could have used generic list collections or Array or ArrayList, one for input and another as output. But the use of DGVC is the author's preference being in form of spreadsheet that seamlessly display its contents. By including stepwise-workings, we attract more users to our applications and provide much-needed education package.

7. CONCLUSION

Few folks directly impact the society positively and deeply as those who develop software applications to solve human problems. The qualities of Software Developers (SDs) have been discussed and how these impact on their thinking and package development. The accurate results obtained depict that SDs thinking sternly sway their codes and packages, and uncover numerous nature of SDs.

SDs are sure revenue generators. Doing things in diverse ways imply flexibility, which human nature craves for. The Pontes et al. rainfall height methods, as well as the optical models' dynamic link library, work accurately to specifications and the results obtained tally with existing ones. They promise to hasten application development. For the three sites considered, it is in South-Africa sites that PNR is higher than PR. The results from the sites indicate that effective rain height cannot be taken to be constant; there is need for adjustment based on sound experimental results.

More codes are expected to be written, and comparative analysis of the rainfall height models will indicate which one best suit a particular location. Additional models are required in the servers for optics and rainfall height in order to boost the servers capability and usefulness.

8. ACKNOWLEDGMENT

Scripture taken from the New King James Version. Copyright © 1982 by Thomas Nelson, Inc. Used by Permission. All rights reserved.

9. REFERENCES

[1] Adenugba, D.A. (2008). Development and Applications of a Customized Active X Control for Data Entry. J. Res. Sc. Mgt. 6(1) 81-90.

[2] Evangelos Petroutsos (2010). Mastering Microsoft Visual Basic 2010. Wiley Publishing Inc. Pp. 1-1023.

[3] Adenugba, D. A. and Adelusi Temitope Isaac (2011). Code Modularization for Swift Software Package Development. J. Sci. & Ind. Studies. 9(1) 33-38.

[4] John Sonmez (2009). Great Developer Are Librarians. http://simpleprogrammer.com/2009/12/08/great-developer-are-librarians/ Accessed May, 2014.

[5] John Sonmez (2012). The 4 Most Important Skills for a Software Developer. http://simpleprogrammer.com /2012/12/09/the-4-most-important-skills-for-a-software-developer/ Accessed June 2014.

[6] John Sonmez (2013). What Software Developers Can Learn From Weiner. http://simpleprogrammer. com/ 2013/07/29/what-software-developers-can-learn-from-weiner/ Accessed July 2014.

[7] Andy Lester (2014). Seven Things You should Never Code Yourself. http://blog.newrelic.com/ 2014/07/08/7-things-never-code/ Accessed July 2014.

[8] Gabriel Zowan (2012). The Forces Against Reform. The Guardian Monday October 29, 2012, Back page.

[9] www.celebritynetworth.com/articles/entertainment-articles/25-richest-people-lived-inflation-adjusted/

[10] Asmita Bhattacharyya and Bhola Nath (2011). Women in Information and Communications Technology. Asian J. Science and Technology. 2(3). Pp. 006-014.

[11] CSTA Voice. Vol.8. Issue 4. September 2012 p.7.

[12] www.bls.gov/ooh/computer-and-information-technology/home.htm

[13] Tim Patrick (2008). Programming Visual Basic 2008. O'Reilly Media, Inc. Pp. 1-725.

[14] Adenugba, D.A, Ojo, J.S and Balogun, E.E, (2010). Development Of A Customized Computer Software Package For Rain Height Evaluation. J. of Inst. of Mathematics & Computer Science (Computer Science Ser.). 21(2). 141-152.

[15] Ajayi, G.O, Feng S, Radicella, S.M and Reddy, B.M. (1996). Handbook On Radio Propagation Related To

Satellite In Tropical And Subtropical Countries. Pp.3-55, 140-186.

[16] ITU-R; Recommendation ITU-R, Rain Height Model for Prediction Methods. p. 839-3.

[17] Pontes, M.S; L.A.R Silva Mello and R.S.L Souza (1994). Radiometric Measurements Of Effective Rain Height, CLIMPARA, Moscow, Russia.

[18] Jay Orear (1979). Optics. In: Physics. Macmillan Publishing Co., Inc. Collier Macmillan Publishers.

[19] Nelkon, M (1978). Optics Section In Principles of Physics. CSS Bookshops and Hart-Davis Educational. 7th ed. Pp. 257-335.

[20] http://en.wikipedia.org/wiki/Brewster%27s_angle. Accessed June, 2014.

[21] http://www.princeton.edu/~achaney/tmve/wiki100k/docs/Brewster_s_angle.html Accessed June, 2014.

[22] http://hyperphysics.phy-astr.gsu.edu/hbase/phyopt/polref.html. Accessed July, 2014.

[23] http://vallance.chem.ox.ac.uk/pdfs/ReflectionRefraction.pdf. Accessed July, 2014.

[24] http://www.physnet.org/modules/pdf_modules/m225.pdf. Accessed August, 2014.

[25] Vahid Tayefeh (2013). Experiment 11: Dispersion From a Prism and Index of Refraction. http://www.academia.edu/5154601/Experiment_11. Accessed August, 2014.

[26] Assis, M.S (1993). Some Remarks On The Spatial Structure Of Rain In Tropical And Equatorial Regions. ISRP'93, Beijing, China.

[27] ITU-R 838-2 (2003). Specific Attenuation Model For Rain For Use In Prediction Methods. Pp.1-5.

[28] Kubista.E, I.P.V Baptista, W.L Randeu AndW.Riedler (1990), Preliminary Evaluation Of Vertical Rain Structure Using Frequency- Agile Dual Polarization Radar, Rio. Pp. 137-142.

APPENDIX A

```
   ''' <summary>
   '''  Computes Multiple Results for Snell's Law.
   ''' </summary>
   ''' <param name="DGVIn"></param>
   ''' <returns></returns>
   ''' <remarks>Row 1 not read being for data title, so start
reading from row 2 zero-based(2-1=1)</remarks>
   Private Function dvCompSnellLaw(ByRef DGVIn As
DataGridView, ByVal dvReportBack As Boolean) As
String
      Try
For gr As Integer = 1 To DGVIn.RowCount – 2    'Less 2: 1
for fixed Row and 1 for Title Row.
 'Read Inputs to Use. Column and Row are zero-based.
   dvValu1 = DGVIn.Item(0, gr).Value  'i
   dvValu2 = DGVIn.Item(1, gr).Value  'r
 'Checks Inputs, reject negative values. Computes Result and
Returns it.
dvResult = dvCalOpticsModels(EnumOptics.SnellLaw)
'Store Result in a reference DGVControl. dvResultColl is
gotten from the method that calls this function.
DGVIn.Item(dvResultColl, gr).Value = dvResult
Next gr
      Catch ex As Exception
'vbCrLf = Carry Return Line Feed. Moves it to the next line
dmsg = ex.Message.ToString & vbCrLf & "Ref:...
dvCompSnellLaw"
         'Display Message on Request from user, if
         dvReportBack = true, else don't report.
         If dvReportBack Then MsgBox
         (dmsg, MsgBoxStyle.Critical,
         "Computing Snell's Law")
         Return "NR"
      End Try
      Return dcomm  'dcomm = Okay; indicates
      that operation is successful. Author's style.
   End Function
```