Simplification of Arithmetic and Variable Script in Hypertext Preprocessor (PHP)

Danial Kafi Ahmad
INTI International University
8 PUBLICATIONS **2** CITATIONS

Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 7.056

IJCSMC, Vol. 9, Issue. 5, May 2020, pg.21 – 34

Simplification of Arithmetic and Variable Script in Hypertext Preprocessor (PHP)

Danial Kafi Ahmad[1,2,*]
[1]Faculty of Information Technology, INTI International University, Nilai, Malaysia
[2]Language Centre, National Defence University of Malaysia, Kuala Lumpur, Malaysia
*Corresponding Author: danialkafi.ahmad@newinti.edu.my, 3181121@alfateh@upnm.edu.my

Abstract- Programming had become as one of the vital skills required in Computer Science, Information Technology, Space Science, Engineering and their related field. This could be seen through the computing industry specifically in the software development area whereby a software based systems were developed in order to perform or cater a specific task by mean of automation, and this will require a developer to deploy their programming skills, ability and experience in order to develop the system. Despite of the functioning software system, their efficiency also take place as one of the important factor which contribute to the quality of the software produced. In this research, a few lines of code which comprise of variables, arithmetic operator as well as their operands of Hypertext Preprocessor (PHP) were developed and were simplified into a single line of code. The simplified versions of code were able to perform the similar computation as to the initial design. In addition, the single line of code is assumed to be more efficient than code of few lines.

I. INTRODUCTION

Software Development had become as an important field in today's modern world. For example, the software based system were deployed in many organizations and fields such as banking institution, education, trading, transportation etc. [1]. This is due to the nature of the software based system which has the ability to provide automation transaction which resulted in more accurate output and fast processing time. However, the system may be developed by different programming or scripting languages, developers of different skills and experience. Therefore, a system could be developed and deployed to perform a transaction, however the efficiency i.e. processing time etc. could be affected by the system design which is the code or script design. For example, in a case of a program which require to print out few lines of alphabets with line break will definitely require a repetition of the program itself. So in this case, the code may be duplicated and executed to get the output or the control structure of looping may be implemented. At most of the time, the usage of the looping is essential as it provide more practical and efficient code flow [2]. In addition, the number of lines of code plays a vital role in the code design whereby it affected the temporary memory consumption of the computer machine. For example, the more lines of codes, the more executions and memory consumption. Therefore, an experiment of simplifying the codes seems vital in order to understand the benefits of code simplification which affect the execution efficiency. Apart from the software manufacturing industry, code simplification also is as important as it provide an important concept in programming in education. For example, the code simplification exercise would let the students to think about how those variables, arithmetic operands and operators work in the codes in terms of the flow and concept. If the students does not understand how those works and their main concepts, therefore they would face difficulties when simplifying the codes. The ability to simplify the codes reflect the high understanding of the code concept, in specific the concept of value parsing and sequence of execution in programming. Thus, by possess the high understanding of the codes, this would lead to creation of software based system with better quality in terms of execution efficiency. As Hypertext Preprocessor (PHP) is one of the common and popular language used to develop web based application such as E-Commerce and Inventory application as well as easily access by developers and educators, students etc., therefore the language was implemented in this research [3]. In this research, a code snippet which comprises of summation arithmetic operations, variables were developed in Hypertext Preprocessor (PHP) as initial version and the code were simplified into a single line of code which were called as simplified version. The code were tested as to validate the output.

II. METHODOLOGY

SDLC

Software Development Lifecycle (SDLC) was implemented in the development in order to achieve systematic approach which led to more quality software produced. SDLC consist of few stages which are System Planning and Selection, Analysis, Design and implementation with testing. These phases were crucial in software development especially for large scale project. SDLC allowed the activity of getting the system requirement, classifying the system functionalities, writing codes or scripts, installation and configuration, as well as testing and maintenance. Figure 1 shows the model of SDLC with their phases.

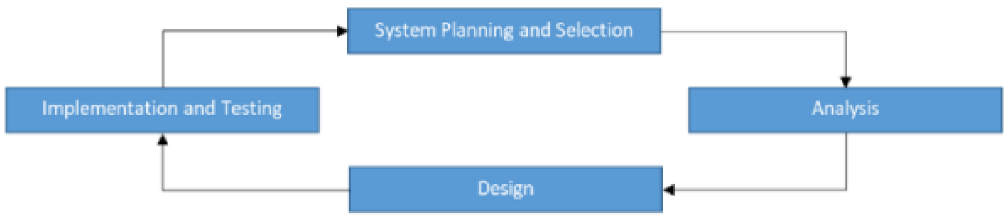


Figure 1. System Development Lifecycle (SDLC). [4]

However, the activities in SDLC would be executed and perceived in different manner, whereby some of the developers might execute the phases in fixed sequence and some of the developers may execute the phases in non-fixed sequence manners. This led to the creation of Plan-Driven and Non-Plan Driven concept.

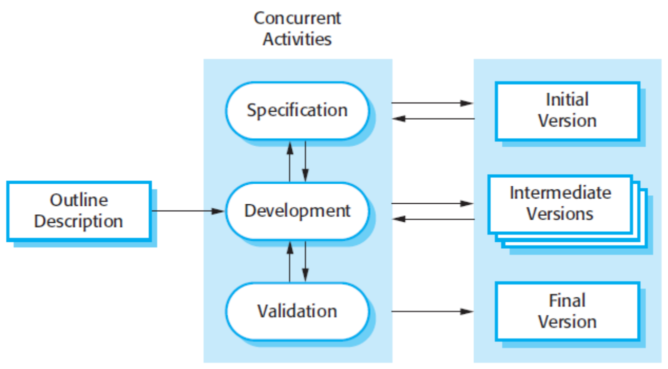


Figure 2. Incremental Model. [4]

Plan Driven is the process of software development in which the activities in within the processes are fixed in term of executions' sequence, while Non-Plan Driven or agile promote an ad-hoc sequence of execution of activities in the model. For example, the waterfall process which comprise of Requirement, Design, Implementation and Unit Testing, Integration and System Testing, as well as Operation and Maintenance were executed from top to down (diagonally) in sequence as in literally as how river waterfall works, without going back to the previous stage [5]. However in this research, an incremental model of agile approach were implemented in order to develop the program. This is due to the nature of the program which were declared as a small scale software based project. Research found that Agile approach more suits to small project, however does not mean that it is not deployable on large scale project [6]. Figure 2 shows the incremental model of software development.

The model allowed the developer to gain more flexibilities during software development whereby the phases of the model which are Specification, Development, and Validation were executable interchangeably among each other where necessary. The model would produce system requirement and design as an initial version, the complete program and tested complete program as intermediate version and final version respectively.

Software Process Model (Incremental)

Specification

Development Tools

The Web-based program were written in PHP and HTML scripting language due to its easy access and high usability. The Apache Server is used in order to run the PHP scripts along with the HTML scripts. Configurations take place as where necessary towards the port of the Apache in the `httpd.conf`. In addition, the changed port which is 8080 were written into the Unified Resource Locator (URL) when requesting page from the server. PHP and HTML were differ in which they perform computational of functional requirement and graphical user interface respectively. Both of the language possess high ability of functionality [7].

Architecture-Client Server

The program implemented Client Server Architecture whereby it requires a client (Browser platform) to request the page from the server. "Currently, the client-server technology is widely used in modern society. The main idea of this concept is that the client sends a request to the server, and the server, in its turn, gives a result" [8]. However, for this project, the localhost server (XAMPP package) were deployed to allow interconnected between components (browser and localhost) in within the same machine.

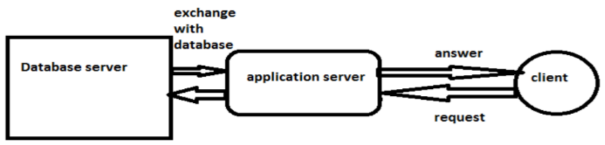


Figure 3. Three Tier-Client Server Architecture [8].

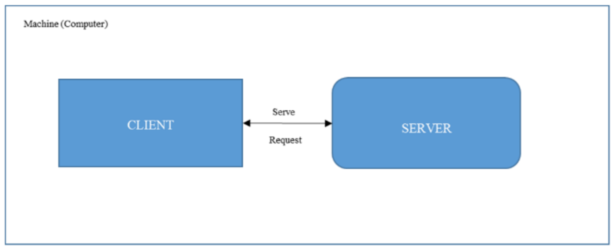


Figure 4. Client Server Component (localhost)

Pseudocode

Pseudocode were used in this research in order to depict the structure and flow of the program [9]. In any software development, pseudocode is considered as a pre-code whereby a coding like elements which consist of program structure and logical flow or semantic flow. The different among the code is that pseudocode is not executable via automated program, and therefore it does not affected by syntax error. In other words, pseudocode is considered as less sensitive in terms of the syntax as it does not take the script symbols but rather the sequence or flow of the execution. In addition, it does not imply a specific rules in terms of syntax but rather the general idea of a program. In testing, the pseudocode is used in order for the testers to analyze the software or program flow, and this is similar to any development process of design or specification phase whereby pseudocode were used before the coding in implementation. In this research, two versions of program were developed which are the initial version whose program were not simplified and were in a few lines whilst the simplified versions were the code form by a single line. Figure 5 and figure 6 shows the pseudocode of initial version and simplified version respectively.

```
BEGIN

       Declare width=3cm and length=4cm

       Perform width * length and store the sum value in area

       Display string of "width * length", the area, string of subscript

END
```

Figure 5. Pseudocode of initial version.

```
//The first version

BEGIN

Display variables of width and length followed by the string of "*","cm","=", area variable, string of "cm" and
subscript

END

//The second version

BEGIN

Display variables of width and length followed by the string of "*","=", area variable, string of "cm" and subscript

END
```

Figure 6. Pseudocode of simplified versions.

Flow Chart

Modelling in software development had become essential and common due to the fact that the model is able to express at least a general or perhaps more specific idea of the system flow. In this research experiment, the flow chart concept and information were deployed indirectly by the researcher to understand the program flow. In common practice of software development, flow chart were developed before the implementation in order get the idea of the system flow [10]. Table 2 shows the components of flow chart which are arrow (flow), diamond shape (condition), rectangle shape (statement/execution), trapezium (input/output).

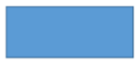
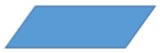
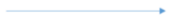
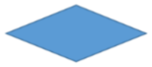
Shape	Action
	**Statement of Execution**
	**Input or Output**
	**Flow**
	**Decision**

Table 2. Flow Chart Diagram Components. [11]

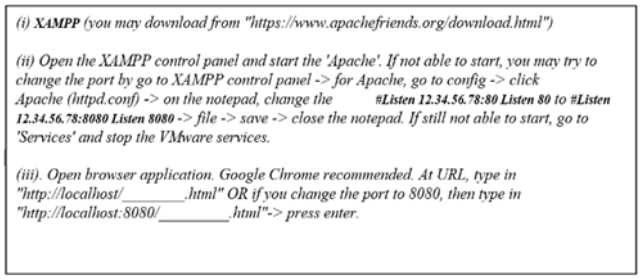
Development

Code

The program used variable, arithmetic operators with their operands and html tag of subscript which written in a form of PHP. The variables used in the programs were `$width="3cm" length="4cm"` and `$area=$width*$length`. The data type used in this program were the integer type and string types. Concatenation (.) were used to combine the variables and the strings to form a sentence. Multiplication (*) were used to perform the calculation of `$area` and the display statement of PHP which is `echo()` were used to print out the output on the screen. The simplification of the program require the usage of precedence concept of arithmetic in order to allow the initial version to be converted to simplified versions.

Installation and Configuration

There were few steps which may be required for configuration of the server as show in figure 7, figure 8 and figure 9.

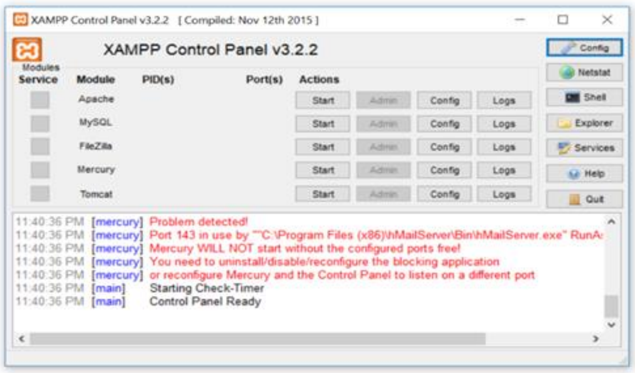


(i) ***XAMPP*** *(you may download from "https://www.apachefriends.org/download.html")*

(ii) Open the XAMPP control panel and start the 'Apache'. If not able to start, you may try to change the port by go to XAMPP control panel -> for Apache, go to config -> click Apache (httpd.conf) -> on the notepad, change the ***#Listen 12.34.56.78:80 Listen 80*** *to* ***#Listen 12.34.56.78:8080 Listen 8080*** *-> file -> save -> close the notepad. If still not able to start, go to 'Services' and stop the VMware services.*

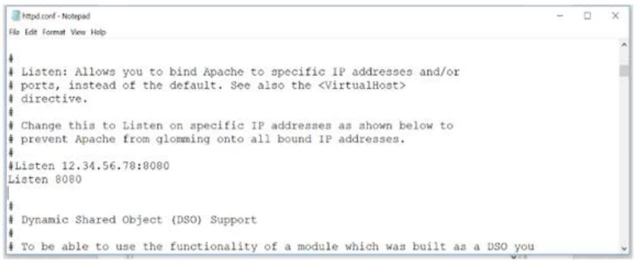
(iii). Open browser application. Google Chrome recommended. At URL, type in "http://localhost/________.html" OR if you change the port to 8080, then type in "http://localhost:8080/_________.html"-> press enter.

Figure 7. Configuration of port via *httpd.conf* file.



```
11:40:36 PM  [mercury]  Problem detected!
11:40:36 PM  [mercury]  Port 143 in use by ""C:\Program Files (x86)\hMailServer\Bin\hMailServer.exe" RunA:
11:40:36 PM  [mercury]  Mercury WILL NOT start without the configured ports free!
11:40:36 PM  [mercury]  You need to uninstall/disable/reconfigure the blocking application
11:40:36 PM  [mercury]  or reconfigure Mercury and the Control Panel to listen on a different port
11:40:36 PM  [main]     Starting Check-Timer
11:40:36 PM  [main]     Control Panel Ready
```

Figure 8. Control Panel of XAMPP Package.



```
#
# Listen: Allows you to bind Apache to specific IP addresses and/or
# ports, instead of the default. See also the <VirtualHost>
# directive.
#
# Change this to Listen on specific IP addresses as shown below to
# prevent Apache from glomming onto all bound IP addresses.
#
#Listen 12.34.56.78:8080
Listen 8080

#
# Dynamic Shared Object (DSO) Support
#
# To be able to use the functionality of a module which was built as a DSO you
```

Figure 9. Configuration of port via httpd.conf script

Validation

Testing were conducted as according to the guidelines of Malaysian Software Testing Board (MSTB) and the International Software Testing Board (ISTQB). The Testing Techniques of White Box Structured and the Black Box Structured were deployed onto the program code in order to see the output and their code's behavior. The test level and type were applied onto both of the techniques in order to achieve the Verification and Validation (V&V) of the program. Rex Black stated that the static technique should be done before the dynamic technique [12].

Test Level

The four levels of testing which are Unit Testing, Integration Testing, System Testing and Acceptance Testing were executed. The following logical model which comprised of *Set {}* were used in visualizing the testing process of each levels. As according to the Susanna S. Epp, the elements or unit can be represented by using the *Set {}* [13].

$$\text{Set U: \{elements\}}$$

$$\text{Let } x \in U, \text{ component}(x) \text{ -> result}(y)$$

Figure 10. Unit Testing Model. [14]

Set U: {elements}

$$\textit{Let } x_b \in U,\ n=<b<=n,\ n=<z<=n,\ component(x_b)\ \ executed \cap component(x_b)\ executed\ \ -> result(y_z)$$

Figure 11. Integration Testing Model. [14]

Set U: {elements}

$$\textit{Let } x_b \in U,\ 1=<b<=5,\ 1=<y<=5$$

$$\frac{(component(xb = 1)\ \ \cap\ component(xb = n))}{((result(y1 = executable)) \cap \text{ result(yn = executable))}}$$

Figure 12. System Testing Model. [14]

Test Type and Technique

In order to achieve the V&V as opposed to the standard Software Development Process. Both Static Technique and the Dynamic Technique, were deployed whereby the program was analyzed with and without automated execution. In addition both techniques were required in the process of testing [15].

III. DISCUSSION

Algorithm

Coding

In this research experiment, there were two versions of program which were developed i.e. the initial version and the simplified version. Figure 13 shows the code of initial version which comprise of variables `$width, $length` which were assigned by an integer of data type of 3 and 4 respectively, and followed by the variable `$area` which was assigned of an arithmetic operations of `$width * $ length`. Those codes or scripts were formed by three lines of codes whereby each of the statement ended by a semicolon (`;`). The last line were about the program printing out the output of those variables after arithmetic computation of variables `$width and $length` executed at line four. For this function, the `echo()` statement of PHP were used in order to display the output to the screen and were concatenated (`.`) with the string of subscript 2 `"<sup>2</sup>"`.

```
<?php
$width="3cm";
$length="4cm";
$area=$width*$length;
echo "$width*$length=".$area."cm"."<sup>2</sup>";
?>
```

Figure 13. Code of initial version.

The code of program of initial version as in Figure 12 was then convert to simplified version which only comprise of a single line which means the statement was ended by a single semicolon (;). This means that the variables with declaration and the arithmetic statement were required to be in one line without changing the meaning or semantic of the program. In this research experiment, the simplified version consisted of only one time variables of `$width and $length`, whereby the declaration occurred on the left hand side of the arithmetic operations. The variables were contained by a pair of brackets `()`, as in, this is due to make sure the priority gave to the declaration on the left Hand Side (LHS), else the variables of `$width and $length` on the right Hand Side (RHS) could not have any values which led to undefined variables. By nature of the program, the variables declaration on LHS will not take place before the execution of RHS, however the rules of precedence created by the brackets () had caused the declaration to occur. Therefore the variables on RHS `($width*$length)` contain values of 3 and 4 of integer data type respectively due to retrieval from the variables on LHS. The computation of `($width*$length)` then were stored in variable `$area`. The `($width*$length)` on RHS were in the brackets as to avoid `the "cm"."<sup>2</sup>"` to be concatenated `with variable $length`, which will cause the string of (cm) and subscript (cm^2) not displayed at the output. Assuming that the brackets were not applied onto `($width*$length) but $width*$length` instead, then `$length` will hold a value of `4cm`^{2}, which then would be computed with multiplication (*) with `$length` which therefore the result of arithmetic computation would hold the value of 12 rather than 12cm^2. This is followed by the value of computation were stored in variable `$area.` In these situation, the usage of brackets had shown the concept of precedence. The simplified versions had shown the same output as the initial version, which reflects the unchanged semantic or meaning of the program. A single mistake in these line of code could lead to major error syntax error and caused the program executed without meeting the correct expectation. The knowledge and understanding towards code abstract, details or so called code quantum were vital in order to able the developer to simplify code where necessary and reduce redundancy.

```
<?php
echo ($width="3")."cm"."*".($length="4")."cm"."=".$area=($width*$length)."cm"."<sup>2</sup>";
?>
```

Figure 14. Code of simplified version1.

However, the simplification could be done in different design whereby the declaration of variables would happen twice and cause more memory consumption of computer. Figure 15 shows the code of simplified version of other design. The `($width=3*$length=4)` require the integer value to be assigned as if not, then the variables would be undefined. For example the variables of `($width=3*$length=4)` if would become this line `($width*$length)` whereby no value assigned, the variables could not take the value of variables on the LHS as the rules is to execute the RHS and stored to the LHS. In this design, there were two time declaration of `$width` and `$length` occur which caused by the removal of brackets of in the line of `echo ($width="3")."cm"."*".($length="4")` and become `echo ($width="3")."cm"."*".($length="4")` on the LHS. Both design in Figure 14 and Figure 15 were form as a single line if code without changing the semantic or meaning of the program. In this research experiment it was found that the first simplified version were more practical due to no redundancy in the codes or scripts.

```
<?php
echo $width="3cm"."*".$length="4cm"."=".$area=($width=3*$length=4)."cm"."<sup>2</sup>";
?>
```

Figure 15. Code of simplified version 2.

Both versions led to the same output as the intention of the research is not to change the meaning and the output but to change the design instead. Figure 16 shows the output of all versions of the program developed.

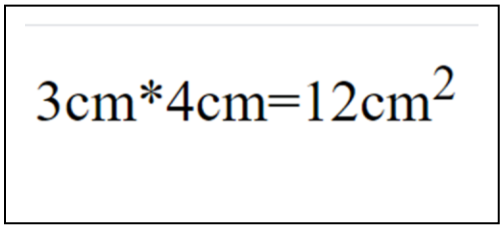

3cm*4cm=12cm^2

Figure 16. Output of the program.

Testing (Validation)

The testing were executed on all of the versions i.e. initial version, simplified version 1 and simplified version 2. Each of the components were tested or debug with the PHP printing statement which is `echo ()` in order to debug the components and view the output. In a system testing, all of the components were combined and tested in single execution in order to view the output. Below figures shows the execution of Unit, Integration and System Testing using the testing model as shown in figure 10, figure 11 and figure 12 which led to Acceptance Testing.

Unit Testing (Initial version)

Set U: { `$width="3cm";`, `$length="4cm";`, `$area=$width*$length;`, `echo "$width*$length=".$area."cm"."<sup>2</sup>";`}

Let $x \in U$, component(x) -> result(y)

component(x= `$width="3cm";`) -> result(y=3cm)

component(x= `$length="4cm";`) -> result(y=4cm)

component(x= `$area=$width*$length;`) -> result(y=12cm^2)

component(x= `echo "$width*$length=".$area."cm"."<sup>2</sup>";`) -> result(y=3cm*4cm=12cm^2)

Figure 17. Unit Testing with components and their results of Initial Version.

Unit Testing (Simplified version 1)

Set U: { `echo ($width="3")."cm"."*".($length="4")."cm"."=";`),

`$area=($width*$length)."cm"."<sup>2</sup>";` }

Let $x \in U$, component(x) -> result(y)

component(x= `echo ($width="3")."cm"."*".($length="4")."cm"."=";`) ---> result(y=3cm*4cm=)

component(x= `echo $area=($width*$length)."cm"."<sup>2</sup>";`---> result(y=O);

*O denote to blank output

Figure 18. Unit Testing with components and their results of Simplified Version 1.

Unit Testing (Simplified version 2)

Set U: { `echo $width="3cm"."*".$length="4cm"."=";` , `$area=($width=3*$length=4)."cm"."<sup>2</sup>";` }

Let x ∈ U, component(x) -> result(y)

component(x= `echo $width="3cm"."*".$length="4cm"."=";`) -> result(y=3cm*4cm=)

component(x= `echo $area=($width=3*$length=4)."cm"."<sup>2</sup>";`) -> result(y=12cm^2);

Figure 19. Unit Testing with components and their results of Simplified Version 2.

Integration and System Testing (Initial version)

Set U: { `$width="3cm";`, `$length="4cm";`, `$area=$width*$length;`, `echo "$width*$length=".$area."cm"."<sup>2</sup>";`}

Let x_b ∈ U

component(x_b) executed ∩ component(x_b) executed -> result(y_z)

component(x= `$width="3cm";`) executed ∩

component(x= `$length="4cm";`) executed ∩

component(x= `$area=$width*$length;`) executed ∩

component(x= `echo "$width*$length=".$area."cm"."<sup>2</sup>";`) executed

-> result(y=3cm*4cm=12cm^2)

Figure 20. Integration Testing model with component executed of Initial Version

$$\frac{\begin{array}{c} component(x = \$width = 3\text{cm};)executed \;\cap\; component(x = \$length = 4\text{cm};)\; executed \;\cap \\ \text{component}(\text{x} = \;\$\text{area} = \$\text{width} * \$\text{length};)\text{ executed } \cap \\ \text{component(x = echo "\$width} * \$\text{length = ". \$area. "cm"." < sup > 2 </sup > ";) executed} \end{array}}{\text{result}(\text{y} = \mathbf{3cm} * \mathbf{4cm} = \mathbf{12cm2})}$$

Figure 21. System Testing model with component as a system executed of Initial Version

Integration and System Testing (Simplified version 1)

Set U: { `echo ($width="3")."cm"."*".($length="4")."cm"."=";)` , `$area=($width*$length)."cm"."<sup>2</sup>";` }

Let $x_b \in U$

component(x_b) executed ∩ component(x_b) executed -> result(y_z)

component(x= `echo ($width="3")."cm"."*".($length="4")."cm"."=";))` executed ∩

component(x= `$area=($width*$length)."cm"."<sup>2</sup>";)` executed ∩

-> result(y=3cm*4cm=12cm^2)

Figure 22. Integration Testing model with component executed of **Simplified Version 1.**

$$\frac{\begin{array}{c}\text{component(x = echo (\$width = "3")."cm"." * ".(\$length = "4")."cm"." = ";))executed } \cap \\ \text{component(x = \$area = (\$width * \$length)."cm"." < sup > 2 </sup > ";) executed } \cap\end{array}}{\text{result(y = 3cm * 4cm = 12cm2)}}$$

Figure 23. System Testing model with component as a system executed of **Simplified Version 1.**

Integration and System Testing (Simplified version 2)

Set U: { `echo $width="3cm"."*".$length="4cm"."=";` , `$area=($width=3*$length=4)."cm"."<sup>2</sup>";`}

Let $x_b \in U$

component(x_b) executed ∩ component(x_b) executed -> result(y_z)

component(x= `echo $width="3cm"."*".$length="4cm"."=";`) ∩

component(x= `echo $area=($width=3*$length=4)."cm"."<sup>2</sup>";`)

-> result(y=12cm^2);

Figure 24. Integration Testing model with component executed of **Simplified Version 2**.

$$\frac{\begin{array}{c} component(x = \ echo\ \$width = \text{3cm.*.}\$length = \text{4cm.=;}) \ \cap \\ \text{component(x = echo \$area = (\$width = 3 * \$length = 4)."cm"." < sup > 2 </sup > ";)}\end{array}}{\text{result(y = 3cm * 4cm = 12cm2)}}$$

Figure 25. System Testing model with component as a system executed of **Simplified Version 2**.

IV. CONCLUSION AND RECOMMENDATION

The code design were mostly affected by the experience and skills of a developer, and the skills were gain from the understanding of the concept. Therefore a good design of code would require a good design from a good developer who own high understanding and skills. In any code or software system of any programming or scripting languages, the efficiency plays an important role in determining the quality of the software. Reducing and avoiding redundancy in coding would affect the memory consumption whereby the computer use less memory to execute or run the line of code or the script. This could be seen more significantly in a very large programs. This conclusion would apply to most of the programming or scripting languages and in this research, PHP were implemented for writing and developing the program due to their easy access as an open source as well as the popularity and the suitability of the language in developing system especially the web based. Therefore, code simplification is very much important and should be practiced and deployed where necessary as it affecting the quality of the software produced.

ACKNOWLEDGEMENT
The main author would like to express his gratitude to the Office of Research Development & Consultancy of INTI International University for the supports in terms of resources allocation for the research and also to Assoc. Prof. Dr. Norshima Zainal Shah, Lt. Kol. Dr. Muhammad Salim bin Muhammad Tufail (B) and Dr. Hoo Yann Seong from the National Defence University of Malaysia for the continuous support to the author in doing research.

REFERENCES

[1] Özer, A.C. and Gürel, H., 2017. Internet Banking Usage Level of Bankers: A Research on Sampling of Turkey. In *Online Banking Security Measures and Data Protection* (pp. 27-39). IGI Global.
[2] Sinha, S., 2020. Flow Control and Looping. In *Quick Start Guide to Dart Programming* (pp. 43-65). Apress, Berkeley, CA
[3] Yadav, N., Rajpoot, D.S. and Dhakad, S.K., 2019, November. LARAVEL: A PHP Framework for E-Commerce Website. In *2019 Fifth International Conference on Image Information Processing (ICIIP)* (pp. 503-508). IEEE.
[4] I. Sommerville, "Software Engineering 9th Edition," Pearson, 2010.
[5] Balaji, S. and Murugaiyan, M.S., 2012. Waterfall vs. V-Model vs. Agile: A comparative study on SDLC. *International Journal of Information Technology and Business Management*, *2*(1), pp.26-30.
[6] Bass, J.M., 2019. Agile on a large scale. *ITNOW*, *61*(1), pp.56-57.
[7] I.F. Amadin, E. Nwelih, "An Empirical Comparison Of: HTML, PHP, COLDFUSION, PERL, ASP .NET, JAVASCRIPT, VBSCRIPT, and PYTON AND JSP. *Global*", *Journal of Computer Science and Technology*, Vol. 10, Issue.12, pp. 9-17, 2010.
[8] D. Mercer, *"Drupal 7,"* Packt publishing Ltd, United Kingdom, 2010.
[9] Hussain, S., Keung, J., Sohail, M.K., Khan, A.A. and Ilahi, M., 2019. Automated framework for classification and selection of software design patterns. *Applied Soft Computing*, *75*, pp.1-20.
[10] U. Gulati, W. Vatanawood, "Transforming Flowchart into Coloured Petri Nets," In Proceedings of the 2019 3rd International Conference on Software and e-Business, pp. 75-80, 2019.
[11] I. Nassi, B. Shneiderman, "Flowchart techniques for structured programming," ACM Sigplan Notices, Vol. 8, Issue.8, pp. 12-26, 1973.
[12] R. Black, D. Graham, E. V. Veenendar, and I. Evans, "Foundations of Software Testing, ISTQB Certification," Cengage Learning EMEA, third edition, pp 35-121, 2012.
[13] S.S. Epps, "Discrete mathematics with applications," Cengage learning, pp. 76-264, 2010.
[14] D.K. Ahmad, M.F. Ahmad, M.N. Ahmad, A.S. Ahmad, "An Experiment of Animation Development in Hypertext Preprocessor (PHP) and Hypertext Markup Language (HTML)," International Journal of Scientific Research in Computer Science and Engineering, Vol.8, Issue.2, pp.45-51, 2020
[15] N. Honest, "Role of Testing in Software Development Life Cycle", International Journal of Computer Sciences and Engineering, Vol.**7**, Issue.5, pp. 886-889, 2019.