Diversity in Utilization of Programming Languages among State Universities and Colleges (SUCs) and Employment Markets in CARAGA Administrative Region

Bryan L. Guibijar
Surigao del Sur State University – San Miguel Campus
Brgy. Carromata, 8301 San Miguel, Surigao del Sur, Philippines
guibijar.16@gmail.com / blguibijar@sdssu.edu.ph

ABSTRACT

The paper focused on determining the extent how the students were taught the programming languages among SUC's and the extent to which the employment market used the programming languages in CARAGA administrative region. The respondents were the IT Teachers and Computer Course Students or IT Students of SUC's and IT Experts from the Industry of CARAGA region. They were made to answer the research-made instruments the contents of which were based on the variables of interest. The gathered data were statistically analyzed using both the descriptive and inferential statistical tools. The hypotheses were tested at a 5% margin of error. The study disclosed that the students were taught less from what the industry needs. The student claimed the highest was on the General Purpose Programming Languages. The findings led to curriculum enhancement and thoroughly maintain monitoring.

Keywords: Programming Languages, IT Instructors/Teachers, IT Students, Computer/IT Curriculum, IT Experts, IT Industry

INTRODUCTION

Determining the fundamental constructs of contemporary and old version programming languages is a vital recipe for new sprout programmers (Sebesta, 2016). In Spain, Computer Science was integrated into K-12 education, due to the positive results obtained in the said research. It was recommended to implement the setting in 5th and 6th grade in primary (Sáez-López, et.al, 2016). There is one report that proposes developing a rigorous undergraduate curriculum for computer science and it intends to model not only for high-quality colleges and universities but also for larger universities with strong computer science programs in a liberal arts setting (Gibbs & Tucker, 1986). This leads to help teachers' introductory programming courses in choosing appropriate first languages and in helping students to overcome the challenges they face (Stefik & Siebert, 2013). The modern programming language industry now has a large variety of incompatible programming languages, each of which with unique syntax, semantics, toolsets, and often their standard libraries, lifetimes, and coasts (Stefik & Hanenberg, October 2014). The study also aims to evaluate the

significant differences between the programming languages taught among SUC's in CARAGA and the programming languages being adopted by the employment market.

Programming is more than just coding, for it exposes students to computational thinking which involves problem-solving using computer science concepts like abstraction and decomposition (Lye & Koh, 2014). The dynamic and reflective features of programming languages are powerful constructs that programmers often mention as extremely useful (Callaú, Robbes, Tanter, & Röthlisberger, 2013). Sometimes debates on programming languages are more religious than scientific (Nanz & Furia, 2015). According to Cass, (2015), a question like "what are the most popular programming languages?" but the only honest answer, depends on what are you trying to land a job at hot mobile app startup, model electricity flows across a continent, or create an electronic art project. SUC's in the Philippines in Region VI integrate a framework for ICT-based development programs for teachers (Magallanes, 2014). According to Bringula, et.al., (2016), very few studies have been conducted in the Philippines to determine if the curriculum sufficiently addresses the needs of the industries.

On the other hand, difficulties faced by the students increase, and demotivation is common in many novice-programming students, who are not able to cope with natural difficulties associated with programming learning (Gomes & Mendes, 2014). The extremely dynamic features like JavaScript make it very difficult to define and detect errors in JavaScript applications (Bae, Cho, Lim, & Ryu, 2014). According to Lister, (2016), he does not claim that it is the only right position, the contrary he alludes to other philosophical position that is regarded as complementary to his own. Many claims and many do not claim, on what is the best programming languages, although others said it depends on the use the programmer itself should consider the updated programming language and the environment that it can be used for running it. In the Philippines, only a few among SUC's are seriously looking at the trends and how suitable a programming language is to a certain course subject.

A curriculum visit will be recommended to address the difficulties encountered by the students. It is relevant to revisit the curriculum especially the subject with laboratories related to computer programming. It should undergo and limit the difficulties by year level and align the programming language used from first to fourth-year level. The programming language to be used will undergo verification if the market needs it. The core foundation of a computer course should be taught aligning it with its originality and applying sustainable and attainable computer courses which are updated on the trends of digital computing.

Theoretical Framework

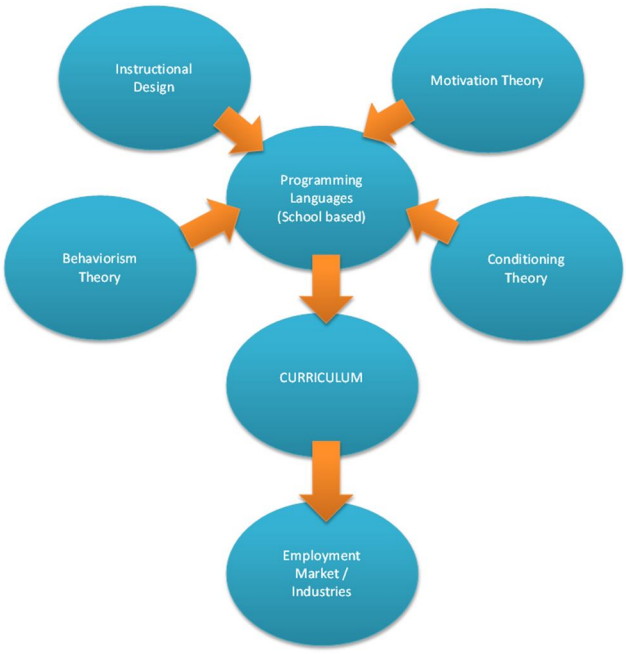

Figure 1. Theoretical Model

The present study underscores the learning theories that prod the formulation of the model of its interest. These theories guided the researcher in the restructuring of existing concepts and the formulation of problems of the present study. This theoretical model is anchored on the four (4) theories of instructional design theory, motivation theory, behaviorism theory, and conditioning theory as elucidated in Figure 1. These theories have their peculiarities when joined together and develop into another educational philosophy leading to the management of technology, which this investigation is designed to formulate.

Conceptual Framework

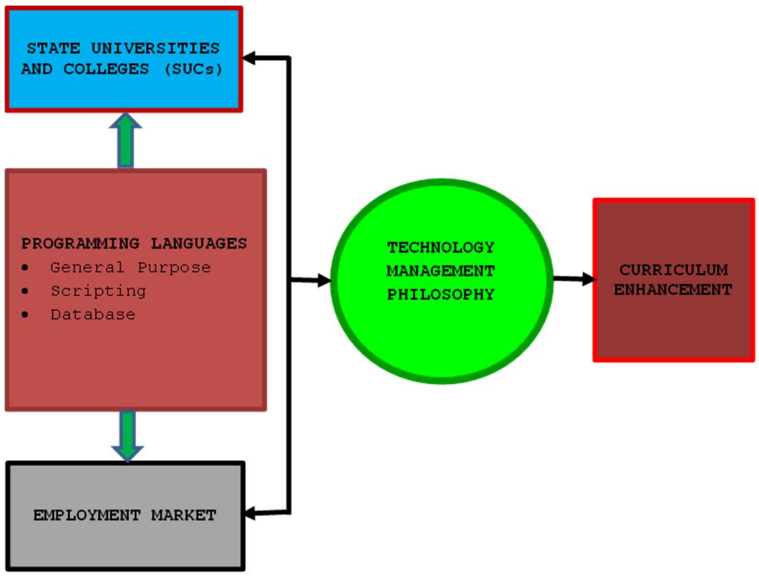

Figure 2. Research Paradigm

The structuring of concepts in this study is introduced in Figure 2. The Figure presents the five frames of concepts that show the interacting forces showing the possible match or mismatch between the programming languages taught in school and also used in the employment markets. The flow of data analysis concentrates on the findings that lead to the formulation of a technology management philosophy that powers the development of curricular enhancement. Technology management philosophy serves as the potent force in the deeper examination of realities in the field based on data and a reference of technological innovation of curriculum.

RESEARCH DESIGN AND METHODS

The descriptive survey research, inferential, and correlation design were used in this study. This was deemed appropriate as the study dealt with the programming languages taught in selected SUC's and the programming languages market demands.

The inferential design employing the differential was used to determine the presence or absence of significant difference among the ratings of respondents in the problems on the programming languages taught in selected SUC's and the programming languages market demands. In this design, the correlation method was used to determine the presence of a significant relationship between the problems on the programming languages taught in selected SUC's and the programming languages market demands.

Research Environment

The study was conducted within the selected SUC's in CARAGA. There are four SUC's in CARAGA located in every province.

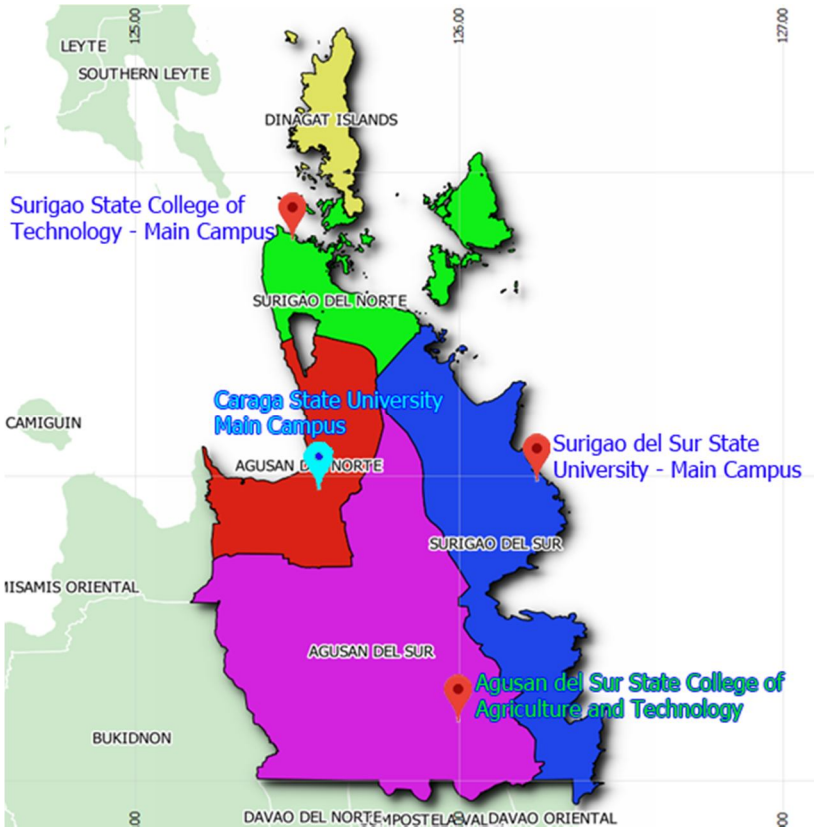

Plate 1. Map of CARAGA

Research Instrument

The researcher-made survey instrument was utilized as a tool for data gathering. There were three sets of instruments, one for the IT teachers (Appendix A), the second for IT Students (Appendix B), and the third will be for the IT heads and workers of the target employment markets (Appendix C). The questionnaire will be composed of three parts.

Part I dealt with information about the source of information as to school and the subject or grade level taught by the respondent. Part II has consisted of items that asked for the use of language programming taught in ICT courses.

Validity. The questionnaires were validated in terms of their content. A draft of the instrument was presented to the adviser and panel of experts for comments and suggestions and the refinement of the said questionnaire. Changes were made and followed, a dry run of the instrument was conducted among selected respondents. With a positive response from the dry run, reliability testing of the instrument followed.

Reliability. This process was initiated after the content validity was established. The researcher employed the run-rerun method where copies of the same instrument were conducted twice to the same respondents observing an hour interval. The reliability was established using the Pearson Product-Moment Correlation Coefficient and the result was shown in Appendix D.

Respondents

The respondents of this study were the students and Instructors/Professors of selected SUC's in CARAGA. In getting the sample size for the students the researcher utilized the lynch law. For the Instructors/Professors, the conventional method was used.

Table 1
Distribution of Respondents

State, Universities and Colleges	IT Teachers	IT Students	IT Heads of Industry/Agency	TOTAL
Surigao State College of Technology - Main (SSCT)	8	93		**101**
Surigao del Sur State University – Main (SDSSU)	8	49		**57**
Caraga State University – Main (CarSU)	4	48		**52**
Agusan del Sur State College of Agriculture and Technology – Bunawan(ASSCAT)	4	60		**64**
CARAGA Region			11	**11**
TOTAL	**24**	**250**	**11**	**285**

Ethics and Data Gathering Procedure

Before the conduct of the study, the researcher asked permission from the President of every SUC in CARAGA (Appendix E). The letters were verified and noted by the Adviser. The request includes the process of establishing validity, reliability, and final conduct of the instrument for the study.

After the letter was approved by the President of the SUC in CARAGA the researcher personally conducted the survey instrument, stressing the purpose of the study to the respondents. After the survey instrument was answered by the respondents, the retrieval, collection, tallying, interpretation, and analysis followed.

Data Analysis

The data were analyzed and interpreted with the following statistical tools:

Weighted Mean and Ordinal rank. These were used to determine the extent to which the students are taught the programming languages and the extent the employment market used the programming languages.

Pearson Product Moment Correlation and **t-test**. These tools were employed to test if there is a significant difference in the application of programming languages and a significant difference between what is taught in State Universities and Colleges and those used in employment markets.

One-Way Analysis of Variance for Correlated Samples. This tool was used to determine the significant difference among the faculty and student extent of teaching the programming languages and the difference in the extent of utilization of the programming languages.

Scheffé Post-Hoc Test. This tool was used to determine which among the factor yielded the significant difference.

RESULTS AND DISCUSSIONS

Table 2
Faculty Ratings on the Extent of Teaching their Students of Programming Languages

Descriptors	Mean	Evaluation	Rank
A. General Purpose			
1. Python	1.46	Low	8
2. C++	2.29	High	2.5
3. Java	2.25	High	4
4. C	2.38	High	1
5. C#	1.96	High	5
6. PHP	2.29	High	2.5
7. Ruby	0.79	Low	11.5
8. Delphi / Object Pascal	0.79	Low	11.5
9. Visual Basic.NET	1.83	High	6
10. Visual Basic	1.79	High	7
11. R	0.71	Low	13
12. Object-C	0.67	Low	15
13. Swift	0.67	Low	15
14. Go	0.67	Low	15
15. Pearl	0.88	Low	10
16. Assembly	1.33	Low	9
Overall Mean	***1.42***	***Low***	***(3)***

B. Scripting			
17. JavaScript	1.83	High	3
18. HTML	2.33	High	1
19. Cascading Style Sheets (CSS)	2.04	High	2
20. Typescript	0.83	Low	4.5
21. Bash/Shell	0.83	Low	4.5
Overall Mean	***1.58***	***High***	***(2)***
C. Database			
22. SQL	2.08	High	1.5
23. PL/SQL	1.54	High	3
24. MySQL	2.08	High	1.5
25. SQLite	1.50	High	4
Overall Mean	***1.80***	***High***	***(1)***

Legend: 2.51-3.00 Very High, 1.50-2.50 High, 0.50-1.49 Low, 0.00-0.49 None at All

The teacher respondents claimed that they taught their students of these general-purpose programming languages at a “low” extent as supported by the obtained overall mean of 1.42. This finding inducts the knowledge that the student receives only very limited knowledge and skills from their teachers on these categories of programming languages.

The overall mean of 1.58 stressed that the IT faculty were teaching their students to a “high” extent. It suggests the knowledge that they have not fully concentrated on teaching the students on the scripting programming languages. Hence, it may be said that there are other priorities that the teachers are giving more importance to than these identified programs.

The IT faculty declared that they taught their students to a “high” extent on these database programming languages as marked by the acquired overall mean of 1.80. This finding signals the knowledge that the IT faculty have not fully maximized efforts in teaching their students these identified programming languages. This further provides an indicator that there are other IT-related activities or tasks that the teachers might have intervened in that captured their focus away from teaching directly the students of these database-related programming languages.

Table 3

Student Ratings on the Extent their IT Faculty Taught them of Programming Languages

Descriptors	Mean	Evaluation	Rank
A. General Purpose			
1. Python	0.90	Low	9
2. C++	1.98	High	3
3. Java	1.81	High	5
4. C	1.99	High	2
5. C#	1.68	High	7
6. PHP	2.09	High	1
7. Ruby	0.62	Low	11.5
8. Delphi / Object Pascal	0.61	Low	13
9. Visual Basic .NET	1.76	High	6
10. Visual Basic	1.93	High	4
11. R	0.49	None at All	16
12. Object-C	0.70	Low	10
13. Swift	0.62	Low	11.5
14. Go	0.57	Low	14.5
15. Pearl	0.57	Low	14.5
16. Assembly	1.10	Low	8
Overall Mean	***1.21***	***Low***	***(3)***
B. Scripting			
17. JavaScript	1.76	High	2
18. HTML	2.11	High	1
19. Cascading Style Sheets (CSS)	1.64	High	3
20. Typescript	0.82	Low	4
21. Bash/Shell	0.74	Low	5
Overall Mean	***1.41***	***Low***	***(2)***
C. Database			
22. SQL	1.99	High	2
23. PL/SQL	1.40	Low	3
24. MySQL	2.19	High	1
25. SQLite	1.03	Low	4
Overall Mean	***1.65***	***High***	***(1)***

Legend: 2.51-3.00 Very High, 1.50-2.50 High, 0.50-1.49 Low, 0.00-0.49 None at All

The student respondents claimed that they have been taught by their instructors of these general-purpose programming languages at a “Low” extent as supported by the obtained overall mean of 1.21. This finding inducts the knowledge that the student receives only very limited knowledge and skills from their teachers on these categories of programming languages.

From the same Table 3, the data yielded information that the IT faculty taught their students the highest on “HTML” with the mean of 2.11, evaluated at “High” extent, and was followed by the “JavaScript” that obtained the mean of 1.76, still evaluated at “High” extent of being taught. The third was a “cascading style sheet” with the mean of 1.64 and was labeled still at a “High” extent of being taught. The lowest taught were on “typescript” and “bash/shell” with the means of 0.82 and 0.74, to stand for “low” extent of being taught.

The students declared that they have been taught by their teacher to a “High” extent on these programming languages as marked by the acquired overall mean of 1.65. This finding signals the knowledge that the IT faculty have not fully maximized efforts in teaching their students these

identified programming languages. This further provides an indicator that there are other IT-related activities or tasks that the teachers might have intervened in that captured their focus away from teaching directly the students of these database-related programming languages.

Table 4
Extent the Employment Markets Utilized the Programming Languages

Descriptors	Mean	Evaluation	Rank
A. General Purpose			
1. Python	1.00	Low	9
2. C++	1.45	Low	5
3. Java	1.64	High	3.5
4. C	1.18	Low	7
5. C#	1.91	High	2
6. PHP	2.71	Very High	1
7. Ruby	0.55	Low	12
8. Delphi / Object Pascal	0.64	Low	10.5
9. Visual Basic .NET	1.64	High	3.5
10. Visual Basic	1.27	Low	6
11. R	0.18	None at All	16
12. Object-C	0.64	Low	10.5
13. Swift	0.36	None at All	13
14. Go	0.27	None at All	14.5
15. Pearl	0.27	None at All	14.5
16. Assembly	1.09	Low	9
Overall Mean	***1.05***	***Low***	***(3)***
B. Scripting			
17. JavaScript	2.82	Very High	1.5
18. HTML	2.82	Very High	1.5
19. Cascading Style Sheets (CSS)	2.73	Very High	3
20. Typescript	0.64	Low	4.5
21. Bash/Shell	0.64	Low	4.5
Overall Mean	***1.93***	***Very High***	***(2)***
C. Database			
22. SQL	2.45	High	2
23. PL/SQL	1.45	Low	3.5
24. MySQL	2.55	Very High	1
25. SQLite	1.45	Low	3.5
Overall Mean	***1.98***	***High***	***(1)***

Legend: 2.51-3.00 Very High, 1.50-2.50 High, 0.50-1.49 Low, 0.00-0.49 None at All

The industry respondents claimed that they utilized the general-purpose programming languages at a “low” extent as supported by the obtained overall mean of 1.05. This finding inducts the knowledge that the industry utilized specific programming for use of their operations.

The overall mean of 1.93 stressed that the industry was utilizing at a “High” extent. It suggests the knowledge that they have not fully concentrated on utilizing all scripting languages. Hence, it may be said that there are other priorities that the industry is giving more importance than these identified languages.

The industry respondents declared that they utilized the database at a “high” extent as marked by the acquired overall mean of 1.98. This finding signals the knowledge that the industry is

not fully utilized every database on their operations. This further provides an indicator that other related ways are utilizing the database in their operations.

Table 5
The difference in the Rating of Faculty and Students on Extent of Teaching Programming Languages

Between Ratings of IT Faculty and Students on: (df = 272 & $t_{.05}$ = 1.96)	**t-value**	**Decision on Ho**	**Conclusion**
1. General Purpose	1.31	Accepted	Not Significant
2. Scripting	0.93	Accepted	Not Significant
3. Database	0.80	Accepted	Not Significant

Revealed in the Table that the ratings of the IT faculty and students on the extent of teaching the programming languages in terms of general-purpose, scripting, and database were not significantly different. The obtained corresponding t-values of 1.31, 0.99, and 0.80 were far below the required critical t.05-value of 1.96 at 272 degrees of freedom. The null hypotheses were all accepted.

The study posited the knowledge that the way the teachers exercise efforts of teaching the various components of IT programming languages are consistent with how their students perceive them. There is a strong agreement between the two groups on the degree of teaching and the likelihood of learning from the IT courses.

Table 6
Difference between the Ratings of Students on Extent of Teaching Programming Languages and Extent of Employment Market Utilization

Between Ratings of IT Students and Employment Sector on: *(df = 259 & $t_{.05}$ = 1.96)*	**t-value**	**Decision on Ho**	**Conclusion**
1. General Purpose	0.73	Accepted	Not Significant
2. Scripting: *($\bar{X}_e$ = 1.93) & ($\bar{X}_s$ = 1.41)*	2.10	***Rejected***	***Significant***
3. Database	1.23	Accepted	Not Significant

Reveled in the Table that the ratings of the IT Students on the extent of teaching and of the Industry on the extent of Employment Market Utilization of programming languages in terms of general-purpose, and database were not significantly different. The obtained corresponding t-values of 0.73 and 1.23 were far below the required critical t.05-value of 1.96 at 259 degrees of freedom. Here, the null hypotheses were accepted. On the other hand, there was a significant difference in their ratings along with the extent of "scripting". The obtained corresponding t-value of 2.10 went beyond the minimal required critical t.05-value 1.96, thus leading to the non-acceptance of the attached null hypotheses.

A deeper analysis of the respective mean of these two groups of respondents, the Employment Market claimed a higher extent ($_e$= *1.93)* of utilization these aspects. The students had mean ratings($_s$= *1.41*) that were remarkably lower than those meant by the employment market.

The study interposes a knowledge that there are areas in the IT infrastructure that need in-depth reconciliatory efforts on the extent of teaching and the degrees of learning which the students acquired. Hence, revisiting the IT curriculum in this aspect in terms of implementation is desired.

TECHNOLOGY MANAGEMENT PHILOSOPHY

Marketable and Employable Curriculum in Computer Courses

"The relevance of Technology is established with a match between learning institutions and employment Markets".

In the first oval (left), the researcher was trying to cite the GAPS in Teaching-Learning between the learning institution and the employment markets. Trends in industries and quality tools in the programming language.

In the second oval (center), the researcher proposed an intervention program to hit the minimum requirements of the employment markets in accepting applicants in their respected offices.

In the third oval (right), the researcher was trying to compensate the researchers' proposed intervention to do curriculum enhancement based on the updates taken from pieces of training and seminar-workshop attended.

Figure 3. Philosophy of Technology Management for Marketable and Employable Curriculum in Computer Courses

Key Result Area	Goal	Objectives	Strategies	Key Performance Indicators	Time Frames				Responsible Person	Source of Fund
					AY 2019-2020		AY 2020-2021			
Curriculum and Instruction	Provide and assured level of education anchored on the demands matches of the industries needs	1. Strengthen the instruction in the following area: cabling; switch; Website; and Open Source	1.1 Conduct in-house trainings and seminar-workshop on Cabling, Switch, website and Open Source	Capacitated faculty, Certificate of Completion of Trainings and Seminar-workshop	1st Semester		1st Semester		VP of Academic Affairs, Dean and Program Chair	Faculty Development
			1.2 Send Faculty to trainings and seminars-workshop more in regional, national and international level.	Skills in Cabling, Switch, Website and Open Source, Certificates of Completion of the Seminar-Workshop		2nd Semester		2nd Semester	Human Resource Officer, VP for Academic Affairs and Dean / Program Chair	Faculty Development
		2. Retool faculty on differentiated instruction or contemporary approaches in teaching and learning Cabling, Switch, Website and	2.1 Determine training needs of faculty in facilitating teaching and learning processes with Cabling, Switch, Website and Open Source	Training needs Analysis basis for the seminar-workshop	1st Semester		1st Semester		Human Resources Officer, VP for Academic Affairs and Dean / Program Chair	Faculty Development
			2.2 Provide seminar-workshop to	Competent faculty, Certificate	1st Sem		1st Sem		Dean / Program Chair	Faculty Development

Open Source especially in planning and preparation of the lesson and assessment of students' learning.	faculty on contemporary approaches in teaching and learning processes with Cabling, Switch, Website and Open Source.	of Completion of the seminar-workshop							Faculty Development
	2.3 Send faculty to relevance seminar-workshop more in regional, national and international level	Proficient faculty, Certificate of Completion of the seminar-workshop		2nd Semester				VP for Academic Affairs and Dean	Faculty Development
	3. Improve delivery of teaching and learning processes with Cabling, Switch, Website and Open Source to match the industry needs with respect to their learning modalities.	3.1 Prepare supervisory plan as a guide in monitoring and evaluating faculty performance	Supervisory Plan	1st Semester				VP for Academic Affairs	Faculty Development
		3.2 Conduct classroom observation based on the supervisory plan.	Classroom Observation Report to monitor effectivenes s	1st Semester	2nd Semester	1st Semester		Dean / Program Chair	Faculty Development
		3.3 Conduct a post conference with the faculty right after classroom	Post Conference Report / Classroom Observation Feed Backing	1st Semester	2nd Semester	1st Semester	2nd Semester	Dean / Program Chair	Faculty Development

		observation for feed backing.			2nd Semester		VP for Academic Affairs and Dean	Faculty Development
	4. Conduct monitoring and evaluation on the effectiveness of the faculty development program.	4.1 Prepare the monitoring and evaluation instrument.	Monitoring and Evaluation Instrument		2nd Semester		Dean / Program Chair	Faculty Development
		4.2 Administer the monitoring and evaluation instrument to the faculty.	Answered Monitoring and Evaluation Instrument		2nd Semester		Dean / Program Chair	Faculty Development
		4.3 Determine the effectiveness of the faculty development program	Result of the Monitoring and Evaluation		2nd Semester		VP for Academic Affairs and Dean	Faculty Development
		4.4 Use the result of monitoring and evaluation for continual improvement.	Recommendations / inputs for the training development program		2nd Semester			

CONCLUSIONS

Teaching appropriate programming languages ready for a job is the best. In general, some findings need to be re-visit and enhanced and there are some also that only need maintain monitoring.

Precisely, the findings of the study led to the following conclusions:

1. The SUC's need to re-visit the curriculum and monitor the teachers if they are following the curriculum syllabus to acquire quality graduates.

2. The SUC's must consider the most used programming languages in the employment market. In the findings it reveals that the SUC's were not totally teaching their students to become more employment in their field, thus the student might work not in-line to the course they graduated.

3. The SUC's must equip their product (course curriculum) to acquire employable graduates.

4. The SUC's just focus on the field of the website to change the significant difference between the way they taught their students and on what is the demand by the employment markets.

RECOMMENDATIONS

Thorough and periodic monitoring of the curriculum and implementing in a syllabus in teaching programming languages.

Explicitly, the following actions are recommended:

School Administrators. They are urged to pattern their leadership to re-visit the curriculum that suits the needs of the industry. Attending seminars, trainings, and other specifics helps the course evolve to make highly competitive graduates.

Professors. As the direct providers of the services to their clients, they are encouraged to be always dynamic, industrious, and goal seekers. They are requested to communicate their needs openly to their superiors on the opportunities for advancement in their lines of work.

Students. They are advised to present their suggestions to the school authority through the client's Suggestion/Feedback box purposely for the improvement of services. They are likewise encouraged to keep open communication with the school personnel to avoid misunderstanding the school policies while studying. Knowledge sharing and feedbacks on trends and experiences are expected from them on matters that may help improve the school.

Researchers. They are challenged to conduct a replicate or another kind of study with a focus on other factors related to the present study about the extent of teaching and the extent of use in the industry. They are encouraged to use the findings as part of their conceptual framework and as a further reference in future investigations.

REFERENCES CITED

Bae, S., Cho, H., Lim, I., & Ryu, S. (2014, November). SAFEWAPI: web API misuse detector for web applications. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 507-517). ACM. Retrieved December 11, 2018. Retrieved from https://goo.gl/2BNQdt

Barr, E. T., Harman, M., McMinn, P., Shahbaz, M., & Yoo, S. (2015). The oracle problem in software testing: A survey. *IEEE transactions on software engineering*, *41*(5), 507-525. Retrieved December 13, 2018. Retrieved from https://goo.gl/vwY1Lq

Benhase, M. T., Gupta, L. M., Mellgren, C. S., & Sanchez, A. E. (2014). *U.S. Patent No. 8,719,529*. Washington, DC: U.S. Patent and Trademark Office. Retrieved December 16, 2018. Retrieved from https://goo.gl/KLa9zJ

Blikstein, P. (2013, April). Multimodal learning analytics. In *Proceedings of the third international conference on learning analytics and knowledge* (pp. 102-106). ACM. Retrieved December 16, 2018. Retrieved from https://goo.gl/cY3Gci

Bracha, G., & Ungar, D. (2015). OOPSLA 2004: mirrors: design principles for meta-level facilities of object-oriented programming languages. *ACM SIGPLAN Notices*, *50*(8), 35-48. Retrieved December 13, 2018. Retrieved from https://goo.gl/prfGjN

Bringula, R. P., Balcoba, A. C., & Basa, R. S. (2016, May). Employable Skills of Information Technology Graduates in the Philippines: Do Industry Practitioners and Educators have the Same View?. In *Proceedings of the 21st Western Canadian Conference on Computing Education* (p. 10). ACM. Retrieved December 06, 2018. Retrieved from https://goo.gl/cEtZ3w

Brown, N. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education (TOCE)*, *14*(2), 9. Retrieved December 13, 2018. Retrieved from https://goo.gl/TdPH3c

Callaú, O., Robbes, R., Tanter, É., & Röthlisberger, D. (2013). How (and why) developers use the dynamic features of programming languages: the case of smalltalk. *Empirical Software Engineering*, *18*(6), 1156-1194. Retrieved December 06, 2018. Retrieved from https://goo.gl/5md9Dr

Cass, S. (2015). The 2015 top ten programming languages. *IEEE Spectrum, July*, *20*. Retrieved December 06, 2018. Retrieved from https://goo.gl/B2njJZ

Chu, H. C. (2014). Potential Negative Effects of Mobile Learning on Students' Learning Achievement and Cognitive Load--A Format Assessment Perspective. *Journal of Educational Technology & Society*, *17*(1). Retrieved December 16, 2018. Retrieved from https://goo.gl/pGSZG9

Dudgeon, K. B., Reed, D. C., Rios, E., & Smith, M. D. (2014). *U.S. Patent No. 8,880,837*. Washington, DC: U.S. Patent and Trademark Office. Retrieved December 16, 2018. Retrieved from https://goo.gl/XJvQ3Q

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education*, *63*, 87-97. Retrieved December 13, 2018. Retrieved from https://goo.gl/DRMT1Z

Flanagan, C., Leino, K. R. M., Lillibridge, M., Nelson, G., Saxe, J. B., & Stata, R. (2013). PLDI 2002: Extended static checking for Java. *ACM Sigplan Notices*, *48*(4S), 22-33. Retrieved December 16, 2018. Retrieved from https://goo.gl/CE67fS

Ghahramani, Z. (2015). Probabilistic machine learning and artificial intelligence. *Nature*, *521*(7553), 452. Retrieved December 13, 2018. Retrieved from https://goo.gl/iwvxaz

Gibbs, N. E., & Tucker, A. B. (1986). A model curriculum for a liberal arts degree in computer science. *Communications of the ACM*, *29*(3), 202-210. Retrieve December 05, 2018. Retrieve from https://goo.gl/yQeRF4

Gomes, A., & Mendes, A. (2014, October). A teacher's view about introductory programming teaching and learning: Difficulties, strategies and motivations. In *Frontiers in Education Conference (FIE), 2014 IEEE* (pp. 1-8). IEEE. Retrieved December 11, 2018. Retrieved from https://goo.gl/3HUwTd

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*(1), 38-43. Retrieved December 16, 2018. Retrieved from https://goo.gl/NQqAAU

Harman, M., Jia, Y., & Zhang, Y. (2015, April). Achievements, open problems and challenges for search based software testing. In *Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on* (pp. 1-12). IEEE. Retrieved December 13, 2018. Retrieved from https://goo.gl/K3t2uN

Ibáñez, M. B., Di-Serio, A., & Delgado-Kloos, C. (2014). Gamification for engaging computer science students in learning activities: A case study. *IEEE Transactions on learning technologies*, *7*(3), 291-301. Retrieved December 16, 2018. Retrieved from https://goo.gl/fBqSWe

Kafai, Y., Fields, D., & Searle, K. (2014). Electronic textiles as disruptive designs: Supporting and challenging maker activities in schools. *Harvard Educational Review*, *84*(4), 532-556. Retrieved December 13, 2018. Retrieved from https://goo.gl/aWcv37

KALELIOĞLU, F., & Gülbahar, Y. (2014). The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective. *Informatics in Education*, *13*(1). Retrieved December 13, 2018. Retrieved from https://goo.gl/XZMsxm

Kazakoff, E. R., Sullivan, A., & Bers, M. U. (2013). The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal*, *41*(4), 245-255. Retrieved December 16, 2018. Retrieved from https://goo.gl/gyafZe

Lamport, L. (2016). Specifying concurrent program modules. *ACM Transactions on Programming Languages and systems*. Retrieved December 13, 2018. Retrieved from https://goo.gl/GqpZPx

Lierler, Y. (2014). Relating constraint answer set programming languages and algorithms. *Artificial Intelligence*, *207*, 1-22. Retrieved December 13, 2018. Retrieved from https://goo.gl/CGfQ5s

Lister, R. (2016, October). Toward a developmental epistemology of computer programming. In *Proceedings of the 11th workshop in primary and secondary computing education*(pp. 5-16). ACM. Retrieved December 13, 2018. Retrieved from https://goo.gl/LcCpya

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12?. *Computers in Human Behavior*, *41*, 51-61. Retrieved December 06, 2018. Retrieved from https://goo.gl/FzBbfn

Magallanes, A. L. (2014). A Framework for an ICT-based Development Program for Science Teachers in State Universities and Colleges in Region VI. *Universal Journal of Educational Research*, *2*(9), 659-668. Retrieved December 06, 2018. Retrieved from https://goo.gl/bNuz6P

Merelli, I., Pérez-Sánchez, H., Gesing, S., & D'Agostino, D. (2014). Managing, analysing, and integrating big data in medical bioinformatics: open problems and future perspectives. *BioMed research international*, *2014*. Retrieved December 13, 2018. Retrieved from https://goo.gl/xQ6cqF

Meyerovich, L. A., & Rabkin, A. S. (2013). Empirical analysis of programming language adoption. *ACM SIGPLAN Notices*, *48*(10), 1-18. Retrieve December 15, 2018. Retrieved from https://goo.gl/WiL17R

Nanz, S., & Furia, C. A. (2015, May). A comparative study of programming languages in Rosetta Code. In *Software Engineering (ICSE), 2015 IEEE/ACM 37th IEEE International Conference on* (Vol. 1, pp. 778-788). IEEE. Retrieve December 06, 2018. Retrieve from https://goo.gl/74nhDV

Park, C. J., & Hyun, J. S. (2014, December). Effects of abstract thinking and familiarity with programming languages on computer programming ability in high schools. In *Teaching, Assessment and Learning (TALE), 2014 International Conference on* (pp. 468-473). IEEE. Retrieve December 13, 2018. Retrieved from https://goo.gl/L7fPXh

Ray, B., Posnett, D., Filkov, V., & Devanbu, P. (2014, November). A large scale study of programming languages and code quality in github. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering* (pp. 155-165). ACM. Retrieved December 13, 2018. Retrieved from https://goo.gl/2BviSQ

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education*, *97*, 129-141. Retrieve December 04, 2018. Retrieve from https://goo.gl/CbN1UY

Sebesta, R. W. (2016). *Concepts of programming languages*. Retrieve December 03, 2018. Retrieve from https://goo.gl/aEPNGv

Shaw, R. S. (2013). The relationships among group size, participation, and performance of programming language learning supported with online forums. *Computers & Education*, *62*, 196-207. Retrieved December 16, 2018. Retrieved from https://goo.gl/7rsHTL

Stefik, A., & Siebert, S. (2013). An empirical investigation into programming language syntax. *ACM Transactions on Computing Education (TOCE)*, *13*(4), 19. Retrieved December 05, 2018. Retrieved from https://goo.gl/Knxg47

Stefik, A., & Hanenberg, S. (2014, October). The programming language wars: Questions and responsibilities for the programming language community. In *Proceedings of the 2014 ACM International Symposium on New Ideas, New Paradigms, and Reflections on Programming & Software* (pp. 283-299). ACM. Retrieved December 05, 2018. Retrieved from https://goo.gl/U1kHWb

Trois, C., Del Fabro, M. D., de Bona, L. C., & Martinello, M. (2016). A survey on SDN programming languages: Toward a taxonomy. *IEEE Communications Surveys & Tutorials*, *18*(4), 2687-2712. Retrieved December 13, 2018. Retrieved from https://goo.gl/V52Rrj

Vee, A. (2013). Understanding computer programming as a literacy. *Literacy in Composition Studies*, *1*(2), 42-64. Retrieved December 13, 2018. Retrieved from https://goo.gl/wJPfr1