European Journal of Economic and Financial Research
ISSN: 2501-9430
ISSN-L: 2501-9430
Available on-line at: http://www.oapub.org/soc

DOI: 10.46827/ejefr.v4i4.1024 Volume 4 | Issue 4 | 2021

EMPIRICAL EVIDENCE ON SYSTEM IMPLEMENTATION FAILURES IN THE ENTERPRISE RESOURCE PLANNING DEVELOPMENT PROCESS

Mustafa Ali Mousa Aljaer[i],
Adam Amril Jaharadak
Post Graduate Centre,
Management and Science University,
University Drive, Off Persiaran Olahraga, Section 13, 40100,
Selangor, Malaysia

Abstract:
Enterprise resource planning (ERP) relates primarily to software for business management, which consists of many modules that are essential for the growth and survival of various types of companies and organisations. By allowing faster supply chain management, the incorporation of similar business applications in an organisation will bring considerable advantages to the organisation. In order to increase business efficiency, ERP implementation helps businesses to adapt and customise information flows and automate business processes. The implementation of the system was reviewed with sufficient literature to substantially support the significance of arithmetic and logical principles and basic programming errors, including programme typographical errors, syntax error detection, and programme output process errors. Furthermore, the declaration of variables, the significance of modularisation and the need to support accessories, the accessibility of library files and the importance of frameworks have been discussed. In order to recognise framework implementation deficiencies in the ERP development process, improvement of coding formatting and significance of coding syntax were also critically checked with appropriate literature.

JEL: L53; O10; L22

Keywords: system implementation, failure, enterprise resource planning, the development process

1. Introduction

For either the implementation team or the organisation, enterprise resource planning (ERP) implementation is not a simple job. It must be carried out with a well-defined

[i] Correspondence: email must7778@gmail.com

strategy by a professional group of technical experts since it is a complicated and costly process to carry out. Although this is a daunting task [1], most companies are required to incorporate enterprise resource planning systems because they can promote the effectiveness and efficiency of current and future corporate business processes [2]. A primary role in the ERP sense is to choose a programming language for implementation and enterprise resource planning customisation does not have a lower value attached to it. This process must be practised with great caution, and there are several factors to be considered, such as the choice of a suitable programming language, the technique of designing and the use of the correct tools [3]. Several variables need to be considered when choosing the right programming language in order to implement enterprise resource planning systems[4]. High-performance computing, web services support and service-oriented architecture (SOA) are the main areas when determining an acceptable programming language [5] [6], considering the ERP capabilities and the rapid technology advertisements.

There are eight subdomains in this component. They are arithmetic and logical concepts (ALC), fundamental errors in system implementation (BESI), programme output process errors (POPE), variables declaration (VD), modularisation (MO), supporting accessories (SA), coding formatting (CF) and coding syntax (CS).

Algorithms are constructed using arithmetic and logical concepts. They are step-by-step finite sequences of instructions to solve a well-defined computational problem. Algorithms are used to solve any complex real-life problems visible in the industry when implementing an enterprise resource planning system. Hence, it is essential to design the algorithm to solve the problem while writing and executing programmes to get the expected output. There are two approaches for algorithm design, namely the top-down and the bottom-up algorithm designs [7].

There are logical principles that need to be met to complete the flow of business while automating business scenarios. All these values must be built correctly. If not, the rate of ERP failure may be high, providing incorrect or unrealistic outputs [7]. Some of the triggers leading to high ERP failure rates, according to Donald's theory [8] are branching performed improperly, loop terminations did not establish well, violated programming language rules and violated programming standards. Programmers also sometimes misinterpret ERP language constructs. Arithmetic algorithm identification overcomes the limitations of some existing traditional methods and adds logical matrices of correlation as a solution to logic diagrams. Besides, a dynamic pruning policy that provides a particular example is executed, as well as complex analyses that separately illustrate the validity and sophistication of the current arithmetic [9]. These concepts are essential to avoid flawed decision logic, arithmetic computations, and erroneous arithmetic computations.

In programming, different types of errors can occur. They are typographical errors such as syntax errors, data errors and programme output errors [10]. These errors are explained in the following sub-sections as follows: 1) Typographical Errors in a Programme [11], 2) Syntax Errors [12], 3) Programme Output Process Errors [13] [14].

The following errors should be remembered and addressed, when necessary, to fix the bugs in the application: 1) identified indexing errors; 2) violated parameters or subscripts; 3) identified data errors; 4) identified non-terminating sub-programmes; 5) identified disk handling errors; 6) identified output processing errors; 7) identified iteratively; 8) procedural errors, and 9) identified initialisation errors.

Document management is vital to the software industry and needs to be maintained well to identify the errors in programming. However, the techniques of document management are applicable with little modification across a wide variety of disciplines, including software. These documents should be modified along with design changes as quickly as possible. Formatting, checking spelling and grammar, organising content and flow of content and also maintaining templates are advisable to make retrieval easy. It should include memos, recipients, and details of senders as well [15]. At its simplest, document management has usually been considered to cover the techniques of creating and or acquiring, storing, locating and retrieving documents throughout their life cycle. However, as the use of computers has increased, documents have increasingly become available. Electronically, the remit of document management has evolved to a point where a "document" can be virtually any sort of computer file, a spreadsheet, a graphics file, a scanned image, a video clip or a voice-mail message [15].

2. Literature Review and Variable Identification

Variables play an essential role in computer programming because they enable programmers to write flexible programmes. Rather than entering data directly into a programme, a programmer can use variables to represent the data. When the programme is executed, the variables are replaced with real data. This makes it possible for the same programme to process different sets of data [16]. Every variable has a name, designated as the variable name, and a data type. A variable data type indicates what sort of value the variable represents. The opposite of a variable is a constant. Constants are values that never change. Because of their inflexibility, constants are used less often than variables in programming [16].

Modularisation programming is a software design technique that emphasises on separating the functionality of a programme into independent, interchangeable modules [17], especially when automating a complex business process like an enterprise resource planning application. It maintains an easy and straightforward way to minimise the error rate of the application. Besides, it gives a clear overview of the application [18].

Nowadays, in programming, there are supporting applications, objects, classes and many more readymade sets of codes that can be incorporated with the existing codes. This is an advantage rather than constructing a programme from scratch. The usability of library files and the importance of the use of frameworks are identified as different types as described in the following sub-sections: 1) Usability of Library Files [19], 2) Use of Frameworks [20] [21] [22].

Coding formatting enhances the readability of lengthy programming and eases the identification of the errors or bugs when fixing bugs. Especially when control structures are used, it is essential to format the code and align them properly by using tabs for indentation, without using both tabs and spaces because not all text editors treat tabs as exactly eight spaces and also leave only one blank line between subroutines to limit blank lines in the programme [23].

The programming language syntax, which forms part of a broader analysis of different programming languages, has been identified to be able to reduce the gap between programming languages [24]. Code syntaxes are the most powerful, which provides the grammar to write a programme. If the syntax is wrong or is misused, the programme will not execute correctly, and it will take two to three minutes to even days to recover, even if a colon or a semicolon is missing. It could become a significant issue or mislead output [14]. The following rules are recommended to be adopted to address this problem: 1) use shift operators instead of multiplication for constructing bit patterns; 2) always check for default case in a switch statement; 3) use each variable for one purpose only; 4) use each field of structure for one purpose only; 5) avoid using global variables within outlines; 6) avoid using nonlocal variables within routines; 7) declare each variable in the smallest scope possible; 8) include all syntax errors to reduce the failure rate; and 9) correct errors promptly as they occur to keep the code simple [23] [25]. In addition, the data visualisation technique also helps to reduce the dimensions of data through the use of self-organising neural networks. Manually generated programming codes as well as automated programming codes show similarities between them and are computed to get a generalised meaning of the syntax trees for the non-vectorial self-organising maps model [22].

Figure 1 shows the research framework for this study. The proposed framework comprises two constructs: (1) enterprise resource planning systems implementation (6) programming language.

The independent variables consist of enterprise resource planning systems implementation. Systems implementation is the construction of the new system and the delivery of that system into output using a computer programme [26].

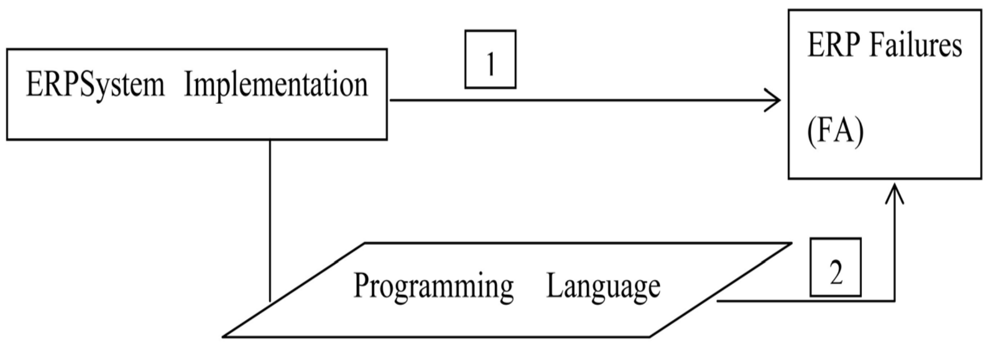

Figure 1: Research framework

The mediating variable is a programming language that can be selected according to the type of application or convenience of the developer [22]. ERP failure rate represents the dependent variable. The proposed research framework broadly depicts the possible relationships connecting the two constructs. The literature was explored to discover the theoretical evidence upon which the hypothetical relationships connecting the constructs were built. These relationships have been planned as a set of research hypotheses addressing the research questions. The following two hypotheses were developed based on the research objectives and research framework:

H1: There is a significant relationship between system implementation and ERP failure.

H2: Programming languages significantly mediate the relationship between system implementation and ERP failure.

3. Materials and Methods

In this research, a quantitative approach was adopted. It is an approach and assessment of a normative sample that is widely used to investigate the views of respondents that represent the entire population. For this analysis, the quantitative approach is more fitting since the goal was to define the factors leading to ERP failure [27]. This approach explains the essence of a condition as it takes place during the time of the analysis and discusses at each and every point of the SDLC the system or structures of a specific condition. Moreover, the technique is acceptable because it helps the researcher to generalise the results obtained [28]. Based on relevant literature, as well as the difficulties and the principles cited by respondents during the pre-survey, the constructs in the questionnaire were established. There were self-administered questionnaires.

In compliance with the Companies Registration Act of Libya, the investigator investigated the enterprise resource planning production companies. However, due to their lack of investment potential and participation in the development of enterprise resource planning applications [29], the software development companies registered under the PLC Act were not considered in this report. Furthermore, when negotiating vendor deals, the PLCs do not have enough money and resources as well as bank guarantees. Therefore, they are not interested in the production of complete solutions for applications for enterprise resource planning. The research unit consists of workers involved in the production of enterprise resource planning applications at each and every level of the implementation of software in software development companies in Libya. By considering staff members with similar educational qualifications and working experience, the respondents were selected using a stratified sampling process. Each member of a population has an equal opportunity under this sampling system to become part of the sample based on the specified developer levels. Since all workers have an equal chance of being study participants, the most effective sampling technique is said to be this sampling method [30]. The researcher obtained a list of all the workers in the sampling strategy and selected the sample for software implementation accordingly. In reality, 66 companies have been registered for BOI software development projects [31]. Only 48 companies fulfilled the inclusion criteria based on the justifications listed in the study environment. There was a total of 3640 workers in all of the 48 enterprise resource planning rises. The representative sample was chosen in the study using a stratified sampling method to pick the workers, followed by applying the random sampling method to distribute the questionnaires. A layer in this study is a subset of the population that shares at least one of the following common characteristics, Software Implementers (SI), with the same academic qualifications for anyone performing the same job position. The sample was used in the programming or creation of the domain for 188 out of the 3640 employees. The survey questionnaire is a research method consisting of a collection of questions and other prompts to collect data from respondents and record their responses.

3.1 Data Analysis

The quality of system implementation is directly related to ERP failure. Accordingly, if system implementation is not being carried out appropriately, this could cause ERP failure and vice-versa. The p-value for ALC, BESI, POPE, VD, MO, SA, CF and CS was less than 0.05. Hence, SI depends on ALC, BESI, POPE, VD, MO, SA, CF and CS. The R-square value was 0.799, which means that 79.9% of the variation in SI is explained by ALC, BESI, POPE, VD, MO, SA, CF and CS. The value of the VIF was less than 5, and hence, there is no problem of multicollinearity. In terms of residual diagnostics, the residuals were independent and normally distributed. The Kolmogorov-Smirnov test of normality on the residuals showed a p-value of 0.049, which is close to 0.05. Thus, the assumption of normality of the residual terms is met. The equation has been constructed as follows:

SI = 0.538 (Constant) + 0.164 (ALC) + 0.135 (BESI) + 0.114 (POPE)+ 0.025 (VD) + 0.155 (MO) + 0.051 (SA) + 0.077 (CF) + 0.122 (CS).

In stepwise regression, only income was significant. The R-square value was 0.999, which means 99.9% of the variation in SI is explained by ALC, BESI, POPE, VD, MO, SA, CF and CS.

SI = 0.004 (Constant) + 0.147(ALC) + 0.206 (BESI) + 0.146 (POPE)+ 0.083 (VD) + 0.085 (MO) + 0.061 (SA) + 0.084 (CF) + 0.186 (CS)

According to the above result, the hypothesis one has explained that there is a significant relationship between system implementation and ERP failure.

The programming languages are posited to have a significant mediating effect on the relationship between SI and ERPF. The score of the Sobel test was 2.4683904 with a significant p-value is 0.0135722, which is less than 0.05. The finding implies that PL is a significant mediator in the relationship between SI and ERPF. Based on the above formula, as is the regression coefficient for the relationship between the independent variable and the mediator, b is the regression coefficient for the relationship between the mediator and the dependent variable, SE a is the standard error of the relationship between the independent variable and the mediator, while SE b is the standard error of the relationship between the mediating variable and the dependent variable. The data was applied as follows:

1) (Unstandardised Beta) $a = 1.000$; $Sa = 0.003$.
2) $b = 0.702$; $Sb = 0.185$.
3) $c = 0.182$; $Sc = 0.056$.
Indirect Effect (IE) $= b \times c = 0.702 \times 0.182 = 0.12776$
Variance in IE $= (b \times Sb)2 + (c \times Sc)2 = (0.702 \times 0.185)2$
$+ (0.182 \times 0.056)2 = 0.016866 + 0.000103 = 0.0169698$
SE in IE
$Z = 0.12776/0.1302 = 0.98112$
The p-value $= P [Z > 0.98112] < 0.001 < 0.05$

Thus, the indirect effect is significant. Using online Sobel application:
Sobel test statistic: 2.4683904. One-tailed probability: 0.006786.
Two-tailed probability: 0.0135722, this is less than 0.05, and therefore PL is a significant mediating factor.

Since the p-value is less than 0.05, PL mediates the relationship between SI and ERPF.

4. Findings and Discussion

The first hypothesis was to determine whether there is a relationship between system implementation and ERP failure. It posited that if the system implementation does not do a proper job, this could be a case of failure to the enterprise resource planning application and vice-versa. Variables representing system implementation such as arithmetic and logical concepts (ALC), fundamental errors in system implementation (BESI), programme output process errors (POPE), variables declaration (VD), modularisation (MO), supporting accessories (SA), coding formatting (CF) and coding syntax (CS) are points of variance that are close to each other in their distribution.

The mean score for system implementation of enterprise resource planning applications was 3.70, with a standard deviation of 0.3792. The maximum and minimum scores were 4.58 and 2.83, respectively. The median was 3.80, which indicates that at least 50% of the system developers graded more than 3.80. Thus, the most frequent rating amongst the system developers was 4.00. This shows that the factors identified in the system implementation questionnaire for ERP system design are significant.

The correlation coefficient for items for each factor, such as ALC, BESI, POPE, VD, MO, SA, CF, and CS, was represented by p-values as low as 0.028, which are less than

0.05. Thus, the items representing system implementation are significant predictors of ERP failures.

For every unit increase in ALC, ERP failure is expected to drop by 0.171. For every unit increase in BESI, ERP failure is expected to drop by 0.144, and for every unit increase in POPE, ERP failure is expected to drop by 0.139. For every unit increase in VD, ERP failure is expected to drop by 0.052. Furthermore, for every unit increase in MO, ERP failure is expected to drop by 0.110, and for every unit increase in SA, ERP failure is expected to drop by 0.067. In addition, for every unit increase in CF, ERP failure is expected to drop by 0.057, and for every unit increase in CS, ERP failure is expected to drop by 0.161.

According to the regression analysis, ALC, BESI, POPE, VD, MO, SA, CF, and CS are the significant predictors that support the objective, i.e. there is a significant relationship between system implementation and ERP failure, and the relationship is in an inverse direction. This implies that the more these factors are being practised; the lower would be the ERP failure rate.

The VIF values for ALC, BESI, POPE, VD, MO, SA, CF and CS are 1.847, 2.257, 1.637, 1.282, 1.062, 1.733, 1.700 and 1.042, respectively. Since they are be- low 5, there is no problem with multicollinearity among these factors.

The findings show that arithmetic and logical concepts should take into consideration that flawed decision logic, arithmetic computations, branching performed incorrectly, loop terminations undefined, violated programming language rules and standards, as well as ERP misinterpreted language constructs by the programmers, are the leading causes that directly affect ERP failure [7].

In addition, this study has also confirmed that basic errors in system implementation include typographical, syntax, indexing, data, disk handling, output processing, iterative procedural, initialisation and violated parameters or sub- scripts, as well as non-terminating subprogrammes, are factors that increase the ERP failure rate [10]. Donald [14] also found that programme output process errors play the primary role in order to reduce the ERP failure rate. These include errors such as input-output format, primary storage allocation, identified software interface and identified erroneously "error message" processing, database interface errors, user interface errors [13] and compiler errors.

According to Andrei [11] and [16], the necessity of the declaration of a variable is vital. The variable type and dimensions should not be incorrectly declared while unique names for variables and standard naming methods for library files should be meaningful. In addition to this modularisation, precautions such as including codes within the main() routine, limiting the number of lines in a routine to 50 or less, having subroutines or loops without duplication of codes are also supposed to reduce ERP failure.

Then the value of the following items formatting programming code such as tabs and spaces for indentation, a consistent indentation pattern for a programme's control structure, and limiting blank lines in programmes, which were identified by Chris [23] and was also confirmed by this research. In addition, the researcher also found that coding syntax with the use of shift operators instead of multiplication for constructing bit patterns, switch statement, check for default case, using each variable for exactly one purpose, avoiding using global variables within routines and nonlocal variables within routines, declaring each variable in the smallest scope possible, correcting errors promptly as they occur, as well as keeping code simple were essential to enhance the success of the enterprise resource planning system [12].

The first hypothesis of this research has been examined through H5, which includes eight clusters along with the 48 items. The findings showed that there is a relationship between system implementation and ERP failure. If a proper system implementation is not carried out, then the ERP failure rate will be high. There- fore, to reduce the failure rate, system implementation has to be carried out ac- cording to the research framework proposed in this study.

There are consistent findings as well for this phase of system implementation. According to [32], enterprise resource planning implementation is the ability of the firm to adapt and configure information flows and integrate business processes in order to enhance business performance. Considering the enterprise resource planning back capabilities and the rapid advancements of technology, high-performance computing, web services support, and service-oriented architecture will be the key areas when evaluating a suitable programming language to implement an enterprise resource planning [5] [6]. The method that generates the syntax tree and uses the feature tree to match the knowledge in the syntax tree to identify code knowledge automatically is being widely followed [12]. The experimental results show that the system can effectively and accurately gather the statistic of knowledge in the programme code in real-time.

If the input programme covers the whole range of the language syntax constructs, then the parser corresponding to the generated annotated grammar is able to parse and transform into an Abstract syntax tree (AST) any programme of the given language [11]. It is essential to use code review correctly. Format- ting, spell check and grammar, organising content and flow of content and maintaining templates are advisable. In order to retrieve them, it should include memos and the details of recipients and senders as well [15].

The results also showed that misusing or improper use of arithmetic and logical concepts, including inadequate decision logic, arithmetic computations, branching performed incorrectly, loop terminations undefined, violated programming language rules and, standards, misinterpreted language constructs by the programmer are the leading causes that directly affect the ERP failure [7]. Zhang [9] has pointed out similar results. In this research, more factors have been considered together to determine their impact. That is, again, a major significant contribution to this research.

Fundamental errors in system implementation which include typographical [11], the syntax [12], indexing, data, disk handling, iterative procedural, initialisation and violated parameters or subscripts and also non-terminating sub-programmes are factors that increase the ERP failure rate. Widera [10] has identified the same factors which are needed to enhance the programme. All these scholars have separately explained how to write a programme with the minimum of those errors.

Consistent with Donald [14], this research found that programme output process errors play the primary role in order to reduce the failure rate. This includes the following errors such as input-output format, primary storage allocation, identified software interface and identified erroneously "error message" processing, database interface errors, user interface errors [13] and compiler errors.

According to [11] and [16], a variable declaration with the variable type and dimensions incorrectly declared, unique names for variables used meaningfully and standard naming methods for library files are also imperative. In addition to this modularisation, routine, limit number of lines in a routine to 50 or less, subroutines or loops without duplicate codes also contribute to reducing the ERP failure.

Chris [23] explains that coding, formatting with the tabs and spaces for in-dentation, programme's control structure to follow consistent indentation pattern, and limiting blank lines in the programme were also proved again in this research with regard to all the above factors.

The researcher found that coding syntax with the use of shift operators instead of multiplication for constructing bit patterns, switching statement, checking for default case, using each variable for exactly one purpose, avoiding using global variables within routines and nonlocal variables within routines, declaring each variable in the smallest scope possible, correcting errors promptly, as they occur and finally keeping code simple can enhance the success of the system [12].

The second hypothesis was to determine whether programming languages (PL) mediate the relationship between system implementation and ERP failure. Sobel's test results showed that the value of 2.4683904 with a significant p-value is 0.0135722. This implies that PL is a significant mediator between system implementation (SI) and ERP failure (ERPF).

The findings are in line with the literature based on the two key points supporting PL as a significant mediator. First, modularisation [17], especially when automating a complex business process like an enterprise resource planning application It maintains an easy and straightforward way to minimise the error rate of the application. In addition, it gives a clear overview of the application [18].

On the other hand, Al-Hossan [33] has also identified that limited access to supporting accessories like library files, frameworks, objects, and modules also increase the ERP failure rate among the software development companies.

In the second hypothesis, modularisation which aligns with the codes within the main() routine, limiting the number of lines in a routine or without duplication codes influencing the relation between system implementation using different programming languages is vital to reduce ERP failure [25]. In addition, awareness of supporting accessories such as library files, frameworks, and objectives are also critical, while Zimin [17] and [33] explained similar facts with the limiting factors discussed under the literature review.

5. Conclusions and Recommendation

From a system implementation viewpoint, another interesting suggestion is derived. Logic programming is a sort of paradigm of programming that is primarily based on formal logic. In the business process, it communicates facts and laws regarding certain problem domains. The treatment of arithmetic and logical principles is thus the most significant factor when automating the application of enterprise resource planning. Knowing the full spectrum of language syntax helps to create a programme that is error-free. It was explained that the proper use of code review is significant. It is advisable to format, spell check and grammar, organise text and content flow and maintain templates. It is vital to reduce typographical errors for the creation of abstract syntax trees, syntax errors, indexing, data, disc handling, iterative procedures, initialisation and infringing parameters or subscriptions, as well as non-terminating sub-programs to address bugs in the application in order to minimise the fundamental errors. This researcher has also found some errors in the input-output format to obtain the expected results, critical storage allocation to run the application smoothly with enough RAM and ROM and to store and retrieve data, defined software interface and incorrect "error message" processing to improve user-friendliness, database to store enough data physically or digitally in the cloud, compiler processing to minimise the compiler data.

According to the results of this report, the importance of declaring variables with the variable form and dimensions to represent data correctly was also clarified. Specific names can be used meaningfully for variables. To properly organise the programme and reduce the conflict with system files, standard naming methods for library files are required. Moreover, it is also necessary for modularisation to split a programme's functionality into different, interchangeable modules to make it faster and boost performance.

The researcher clarified that coding formatting with the indentation tabs and spaces improves the coding's readability when referencing them for correction or alteration again. The control structure of the programme advises that clear indentation patterns are followed, and blank lines in programmes are reduced. This was also recommended as a whole, in order to minimise the implementation failures of the programme.

In addition, the researcher found that coding syntax also clarified and recommended the use of shift operators instead of multiplication to create bit patterns, switch statements, use each variable for exactly one purpose, avoid the use of global variables within routines, declare each variable as small as possible and promptly correct errors as they occur, and finally hold them. The syntax is language-specific and varies depending on the language. Only in the sense of a specific language are they essential. Some languages are compiled into another base language, and the syntax is typically different. Depending on what the syntax should be, modifying a language's syntax can be relatively simple or challenging. In order to save time and budget, it is also recommended to know the language accurately.

Selecting a proper and programming language is also very important because day by day technology changes, thereby creating new updates as well as plug-ins and supporting objects. It is essential, recommended and necessary to keep in touch to decide when to quit the existing application and switch to the new generation.

Conflict of Interest Statement

The authors declare no conflicts of interests.

About the Authors

Assoc. Prof. Dr. **Adam Amril Jaharadak** is currently working as the Director (Cyber Security & Big Data Centre) in Management and Science University (MSU) Malaysia. His research and supervisory interest include Information Science, Human-computer Interaction, E-Learning, Information Technology, Information Technology Management, Knowledge Management, Games, Game-Based Learning, Game Studies Evolutionary etc. Furthermore, he has numerous publications in diversified international journals.
Mustafa Ali Mousa Aljaer is currently PhD student (Faculty of Information Sciences and Engineering) at Management and Science University (MSU) Malaysia. He holds a master's degree from the University Technology Malaysia (UTM) in 2006 – 2009. His research interests include Comparison between Transmission Control Protocol Schemes on Wireless Environment. He holds a Bachelor's degree from the College of Engineering in Tajoura, Tripoli, Libya.

References

[1] Samuel, A.A., Reddy, B.S. and Nair, J. (2014). Conceptualising Dimensions of Enterprise Resource Planning Systems Success: A Socio-Technical Perspective. International Journal of Enterprise Information Systems, 10, 53-75.

[2] Shivkumar, S. and Hasmukhrai, T. (2012). Software Testing Techniques. International Journal of Advanced Research in Computer Science and Software Engineering, 7, 16.

[3] Howden, W. (2006). Reliability of the Path Analysis Testing Strategy. IEEE Transactions on Software Engineering, SE-2, 208-215. https://doi.org/10.1109/TSE.1976.233816

[4] Chaim, M., Maldonado, J. and Jino, M. (2003). A Debugging Strategy Based on Requirements of Testing. Seventh European Conference on Software Maintenance and Reengineering, Benevento, Italy, 28 March 2003, 160-169.

[5] Galin, D. (2004). Software Quality Assurance. Pearson Education Limited, Harlow.

[6] Lewis, W.E. (2005). Software Testing and Continuous Quality Improvement. Auer-beach Publications, New York.

[7] Vinu, V.D. (2006). Principles of Data Structures Using C and C++. New Age International, Telangana, India.

[8] Donald, E.K. (1995). The Art of Computer Programming. Vol. 3, Odd Bookworm, Hackensack, NJ.

[9] Zhang, L., Yuan, S., Tang, J. and Xie, X. (2008). Research on the Composite Arithmetic of Logic Compound Sentences in Decompilation. 2008 International Symposium on Computer Science and Computational Technology, Shanghai, 20-22 December 2008, 442-446.

[10] Widera, M. (2011). Why Testing Matters in Functional Programming. Fern University at in Hagen, Hagen.

[11] Andrei, A. and Daniel, I.V. (2012). Automating Abstract Syntax Tree Construction for Context Free Grammars. 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 26-29 September 2012, 152-159.

[12] Gayle, L.M. (2011). Cracking the Coding Interview: 150 Programming Questions and Solutions. 5th Edition, CareerCup.

[13] Carlton, R. (2017). Three Dangers of a Poorly-Designed ERP User Interface. Converted Media.

[14] Knuth, D.E. (2007). The Art of Computer Programming. Volume 1, 3rd, Edition, Dorling Kindersley, Delhi.

[15] Doverton, D. (2001). Techniques of Document Management: A Review of Text Retrieval and Related Technologies. Journal of Documentation, 57, 192-217. https://doi.org/10.1108/EUM0000000007082

[16] Vangie, B. (2015). variable.html. http://www.webopedia.com/TERM/V/variable.html

[17] Jin, Z.M., Fu, Q., Jin, J. and Tao, J.W. (2013). Characteristics and Module Design of Weaving ERP. 3rd International Conference on Information Management, Innovation Management and Industrial Engineering, Kunming, 26-28 November 2010, 422-425.

[18] Kenneth, E. (2011). Modularisation. Aalborg University, Copenhagen.

[19] Georges, E.K. (2009). Building a Service-Oriented ERP from an Open Source Software. Fourth International Conference on Software Engineering Advances, Porto, Portugal, 20-25 September 2009, 33-38.

[20] Leopoulos, V., Kirytopoulos, K. and Voulgaridou, D. (2005). ERP Systems as a Component of the Electronic Supply Chain: Classification of Implementation Risks. International Conference on Computational Intelligence for Modelling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, Vienna, Austria, 28-30 November 2005, 676-682.

[21] Song, H., Huiyou, C. and Qing, W. (2009). Component Library-Based ERP Software Development Methodology. 2009 International Conference on Interoperability for Enterprise Software and Applications China, Beijing, 21-22 April 2009, 34-38.

[22] Zhu, Z.Y. and Dai, S.H. (2009). J2EE-Based Enterprise ERP System Design and Implementation. 2009 2nd IEEE International Conference on Computer Science and Information Technology, Beijing, 8-11 August 2009, 509-512.

[23] Chris, P. (2009). Learn to Program. 2nd Edition, Pragmatic Bookshelf.

[24] Andreas, S. (2013). An Empirical Investigation into Programming Language Syntax. ACM Transactions on Computing Education, 13, Article No. 19. https://doi.org/10.1145/2534973

[25] Samantha, M.A, Chong, S.C. and Kennedy, D.G. (2014). A Comparison between Evaluation of Computer Based Testing and Paper Based Testing for Subjects in Computer Programming. International Journal of Software Engineering and Applications, 5, 57-72. https://doi.org/10.5121/ijsea.2014.5105

[26] MITRE (2013). System Design and Development. MITRE Corporation, Bedford, MA.

[27] Creswell, J. (1994). Research Design: Qualitative and Quantitative Approaches. Sage Publications, New York.

[28] George, D. and Mallery, P. (2013). IBM SPSS Statistics 21 Step by Step: A Simple Guide and Reference. 13th Ed, Pearson, Upper Saddle River, NJ.

[29] Central Bank of Sri Lanka (2014). Annual Report 2014.

[30] Sekaran, U. and Bougie, R. (2013). Research Methods for Business A Skill-Building Approach. 6th Edition, Wiley, New York.

[31] Board of Investment (2012). BOI. http://www.investsrilanka.com/

[32] Srinivasan, D.D. and Gopalaswamy, R. (2006). Software Testing: Principles and Practice. Pearson Education, India.

[33] Al-Hossan, A. and Al-Mudimigh, A.S. (2011). Practical Guidelines for Successful ERP Testing. Theoretical and Applied Information Technology, 27, 11-18.