PERCEPTIONS OF SOFTWARE DEVELOPERS ON THE USE OF SOFTWARE FRAMEWORKS TO DEVELOP INFORMATION SYSTEMS (I.S)

SHIYNSA Charles Lwanga[1]
Research scholar, School of ICT, ICT-University Cameroon
shiynsacharlesl@yahoo.com

ABDALLAH ziraba[2]
Senior lecturer ICT-faculty, ICT-University
abdallah.ziraba@ictuniversity.edu.cm

SHIYNSA Divine Ghakanyuy[3]
Research scholar, CITEC-HITM, Shell Nsemiyong, Yaounde
Shiynsadecallo40@yahoo.com

ABSTRACT

Developers have demonstrated how quick and affordable it is to develop information systems (I.S) using software frameworks. It should be noted that software developed using frameworks are likely to fail. Based on previous studies carried out on 156 well scrutinized research articles using a mapping methodology to articulate on software failure related problems, majority of the articles confirm the fact that software related problems are caused by source codes of frameworks. The research method used was qualitative. The researchers had an in-depth interview each with 15 software developers in Cameroon that have used software frameworks for at least 5 years. Grounded theory as a qualitative data analysis technique was used and triangulation to justify reliability and validity was adopted. The results revealed that hidden cost and coding skills associated with dependencies like plugins contributes to failing of software/I.S. The findings also revealed that premium dependencies and training is needed to build systems that meet objectives. This study recommends that the developers should cease from using null dependencies, continuously improve on coding skills, install system updates, encourage team work, embark on feasibility study before building systems and test software using different testing techniques to close loopholes.

Keywords: I.S; software; plugin; Software framework; I.S developer.

INTRODUCTION

Developers have demonstrated how quick and affordable it is to develop information systems (I.S) using frameworks (Ihor, 2023; Ratnawati et al., 2022) but, it should be noted that software developed using frameworks are likely to fail (Palacios Chavarro et al., 2023). According to Feinbube & Polze (2016), it is emphasized that out of 156 well scrutinized research articles using a mapping methodology to articulate on software failure related problems, a majority of the articles confirm the fact that software related problems are caused by source codes of frameworks and finally, a model was realised to create awareness on the issue of software failure. Additionally, there is a hidden cost associated with developing software/I.S which many developers neglect to account for. According to (Nadu, 2022), the difficulty in finding frameworks with free resources (plugins) to build sophisticated I.S within frameworks architecture and frameworks methods makes it difficult for developers to produce desired products on time.

Previous scholars have explored the perceptions of developers in the use of frameworks in I.S development (Aulia et al., 2022; Bianco et al., 2014; Ihor, 2023; Myllärniemi et al., 2018; Ribeiro & Da Silva, 2012; Susanto et al., 2023; Toman et al., 2019a). The findings revealed a number of useful models developed to explain appropriate development of I.S using frameworks. Unfortunately, the models are not necessarily appropriate in all the situations (Wendler, 2012). According to (Khalid et al., 2014), the developers' omissions to address the hidden costs associated with apps plugins in the development of software/I.S, makes models not always appropriate (Khurshid et al., 2021). As a result of the inability to provide I.S on time, several developers end up losing trust and confidence of the clients.

The aim of this study was to developed a model to guide software developers on how to build software/I. S using frameworks that will satisfy the needs of the client and prevent unforeseen cost that leads to deviation from project objectives. The model is guided by qualitative data gathered to explore developers' opinion and perceptions in utilising frameworks to build software/I.S.

RESEARCH QUESTIONS

To get a better opinion of software developers on the use of software frameworks to build software/I.S, the following questions were asked in a one-on-one interview:

- What are the factors that leads to the failure of I.S/Software built with software frameworks?
- What are the solutions to the problems that leads to the failure of I.S/Software built with software frameworks?

LITERATURE REVIEW

Scholarly Perception of Software Frameworks

Software frameworks or object-oriented frameworks are software codes created by individuals or corporate bodies with underlying algorithms, logic and other resources used by developers to realise cost-effective systems within the shortest time possible to meet system objective. According to Islam et al., (2022), a software framework is an abstraction with possibilities to selectively update software that offer generic functionality by adding a new user-written code. Contrary to Myllärniemi et al., (2018) studies, a software framework is a software with capabilities of providing a reusable architectural design and partial implementation associated with drawbacks like complexity and techniques required to use. Nguyen et al., (2019) has reservations on framework functionalities to build I.S and looks at how extensive, frameworks to implement machine learning algorithms in the area of Big Data handling. Semiawan et al., (2021) are for the opinion that effectiveness and efficiency of frameworks is as a function of the problem the frameworks are used to solve. According to Boell, (2015), I.S built with frameworks constitutes hardware and software interacting to keep actors informed within a given social and organisational context. On the technical perspective, (Laudon C & Laudon P, 2014; Zemmouchi-ghomari et al., 2021) are focal on an I.S to be a set of interacting components aimed at storing, processing, retrieving, and distributing information to get expected feedback. According to Mealin & Murphy-Hill, (2012) Software developers working with frameworks/I.S designs, builds and maintains software. Contrary to Mealin and Murphy-Hill thoughts, (Ayuningtiyas et al., 2023), builds on the fact that should possess skills to permit coding and analysis related task to be cleanly executed.

Perceptions of Software Developers on Using Frameworks

Looking at perception from a software engineering methodology used to build IS, (Nascimento & de Mello, 2022) holds on the opinion; software developers using a given framework to build I.S will automatically determine a development method they have to use. In this context, the project success of a given I.S is a guarantee or not, based on resources (plugins) in the framework and required technical skills to use the framework. Contrary to the arguments of (Mutanga, 2022), security challenges in software development is a backbone of security in I.S. When software is poorly developed by following basic security guidelines without reinforcement using necessary tools, the probability of experiencing security challenges increases. In summary, perception in I.S development is contextual and it sums up to the extent a given developer conceives a system and associates activities to accompany such an I.S.

Scholarly Perspectives on Hidden Cost and Technical Skills associated with software/I. S failures

Several studies with interventions in the use of software frameworks to develop I.S/software have been explored. Hidden cost associated with frameworks that use plugins to build I.S/software is never previewed before the development process starts. According to Semiawan et al., (2021), coders effectiveness and efficiency in using software frameworks to build I.S. does not have a difference with software coded from crash. This is evident in that, software coded from crash and software developed using frameworks are almost the same in terms of usability, completion and quality. In disagreement with Semiawan et al, (Feinbube & Polze, 2016), builds on a thorough scrutiny of about 156 research articles related to software problems. These problems are caused by badly structured and typed codes which leads to a realization of a model to explain causes of software failure. In a mixed method study by Myllärniemi et al., (2018), findings revealed developers have affinity for object-oriented

frameworks with an in-built API which can heavily cut down cost. Nazir, (2022) disagree with Myllärniemi et al by building on the conclusions, developing software and I.S does not forcibly mean using frameworks but also, using cloud resources as noticed in 97 published research works on software coded using cloud-based resources. According to (Nadu, 2022), non-cloud based frameworks having hidden cost. Toman et al., (2019a) conducted a study on the use of web-based frameworks to build I.S and discovered that framework like Electron outperforms NW.js in terms of cost effectiveness. Sharma, (2019) builds on the opinion; to prevent "bad smells" (malfunctioning) in codes built with frameworks, good coding skills, good coding practices and proper structuring of codes is required. Kaur & Singh, (2023) confirms Sharma's findings on "bad smells" of software codes which can cause I.S failure by emphasizing, patching of smelling codes and refactoring due to lack of dependencies is necessary. According to Xie et al., (2013), merging frameworks to have one with more features and capability help in overcoming loopholes or coding extra patches. In a study by (Reshi, 2017), code smells and maintainability help to determine if a software/I.S is failing and needs refactoring (resources). In cognisance of Eclipse and Tomcat related projects, software lapses and failure can be handled by putting in place resources (plugins) and skills. Sharma, (2019) settles on the fact that maintainability is highly required to ensure software built with frameworks are effective. Toman et al., (2019) has reservations on Sharma's opinion while basing on; maintainability which is focused on helping software developers in choosing between frameworks. This will foster trust and loyalty of customers and prevent software failure if the right frameworks are chosen.

METHODOLOGY

This study adopted the qualitative method (Busetto et al., 2020; Dunwoodie et al., 2023) that employs interpretivism epistemology (Avenier & Thomas, 2015) approach. The researchers work with 15 (Nascimento & de Mello, 2022) software developers as respondents by means of a one-on-one interview. Using purposive sampling technique (Dunwoodie et al., 2022; Overconfidence et al., 2023), developers were selected from Cameroon specifically the centre region. Developers were selected based on coding experiences of at least 5 years of continuous coding of I.S/software using frameworks. The distribution of respondents was as follows; 1 male from litoral, 2 males from north west, 1 male from the north, 2 male from the east, 5 male and 1 female from the centre region, 2 male from the west region and 1 male from the south region. The justification for selecting these respondents was based on the fact that these developers have been singled out as developers that have developed I.S systems that have failed between 2020 to 2024. These software developers had an exchange with the researchers for at least an hour each focusing on factors and proposed solutions centered around failing software/I.S built with software frameworks. This interview was conducted using BigBlue. A well-structured interview guide (Dunwoodie et al., 2023) was administered on the respondents. Researchers choose grounded theory (Khan, 2016) as a research design because first class information on frameworks usage to build software/I.S was needed to ensure credible facts is gotten thanks to their experiences and interactions with these frameworks. Focusing on perceptions and attitude of software developers towards frameworks, ontological subjectivism (Ylönen & Aven, 2023) is used. Afterward, open coding, axial coding and selective coding processes were used to code the data into themes. Haven considered researchers to be part of this study, triangulation (Noble & Heale, 2019) was used to justify reliability and validity and after analysis, the researchers contacted the respondents to ensure themes match what they said earlier in the interview. At the end, analysis and interpretation was conducted and reporting was done. Results reveal I.S/software built with software frameworks fail due to hidden cost and insufficient technical skills of

some coders who use software frameworks to build I.S. To overcome this, tech communities, regularly updates of system and continuous training to fine tune coding skills is required.

RESULTS/FINDINGS

This entails the memos and results from the open coding process, axial coding process and selecting coding process.

Memo for coding processes

Fifteen (15) software developers were interviewed from 15 different I.T firms in Cameroon and particularly in Yaounde the capital city to find out their opinions about the use of software frameworks to build I.S/Software as response strategy against failing software/I. S built with frameworks. Purposive sampling was used and developers were asked to give their views and opinions based on experience why most of the systems built with frameworks fail despite the fact that frameworks are the most recommended (Mateen et al., 2017; Toman et al., 2019) to gain time and reduce stress with numerous features and dependencies available. The interviews were transcribed and narratives/codes emerged to explain why I.S/Software developed with software framework fail and proposed solution as a response strategy against failing I.S is presented. Categories such as scalability issues, vulnerabilities, update/upgrade, coding skills, null plugins, feasibility study, compatibility issues, performance and hosting emerged to explain why I.S coded with frameworks still fail despite numerous security attempts put in place. In respond to challenges faced by these failing systems built with frameworks, other categories such as install updates, support document, tech communities, training, system analysis, buy premium, secure, hosting bandwidth and advanced testing techniques about failing I.S built with frameworks are proposed solutions that emerged to explain probable counter mechanisms that can be used to remedy challenges faced by frameworks. The new ideas/ categories that emerged after responses from the first respondent 01 (P001) both causes and proposed solution to failing I.S/software are; feasibility study, compatibility issues, performance, hosting, documentation and hosting bandwidth, advanced testing techniques respectively. After that, from participants 04 (P004) and 08 (P008) respectively, no new ideas emerged leading to a point of saturation and thus, serves as a guarantee that the sample size was valid and appropriate. The memo and refinement of axial coding process entails categories like; scalability issues (some frameworks are not scalable that is subjecting developers to use what is in frameworks without adding new modules, makes developed systems not to meet up objectives), vulnerabilities (this is related to loopholes found in systems built as a result of vulnerabilities found in frameworks that hackers take advantage of if not amended), update/upgrade (this is related to systems that are not frequently updated and are not updatable to meetup with recent security scripts. Thus, causing system to be outdated), coding skills (this is due to lack of required skills to customise the source code of some framework thereby causing more vulnerabilities on the framework), null plugins (this is using free plugins to build systems that are not updatable or upgradable or using free custom-built plugins by hackers harbouring malware codes), feasibility study (this diving into code systems with frameworks without carrying out first-hand enquiries about the software, the framework and what it takes in terms of finance and skills to build the system), compatibility issues (this is incompatibility that occur between the system coded with frameworks having null plugins and the hardware or other software in system), performance (this refer to poor responsiveness like processing speed of the software/I.S coded with frameworks on different gadgets), hosting (this involves the hosting bandwidth used to host the coded system that the speed can subsequently affect performance and codes bugs in the system), install updates (updating of frameworks will curb failure of the framework and subsequently the system), support documentation (this is consulting the framework documentation to check for new updates, new code syntax, new modules or

packages that can be integrated in the I.S/software coded), tech communities (this entails consulting online tech forums seeking for help related to a framework), training (formal training on using framework is required by coders to help them master framework tricks), system analysis (this entails getting detailed information about the system to be built and feasibility studies to help the developers choose wisely the appropriate framework) and buy premium (this is to encourage developers to code systems with premium versions of plugins so as to avoid creating loopholes in systems due to the use of null plugins)

Table 1

Grouping of categories of axial coding process

Question	Categories	Abbr	Frequency	Theme
What are the factors that leads to the failure of I.S/Software built with software frameworks?	scalability issues	SI	2	Non-scalable systems
	vulnerabilities	Vul	11	Software/I.S loopholes
	update/upgrade	UU	11	Regular updates/upgrades
	coding skills	CS	12	inappropriate coding skills
	null plugins	NP	12	Cost-free plugins
	feasibility study	FS	8	Feasiblity studies concerns
	compatibility issues	CI	4	Compatibility problems
	performance	P	6	System functionality
	Hosting	H	8	Hosting complications
What are the solutions to the problems that leads to the failure of I.S/Software built with software frameworks?	install updates	IU	11	updates installation
	support document	SD	8	framework documentation
	tech communities	TC	15	Tech Communities
	Training	T	11	Skillful training
	system analysis	SA	10	Preliminary studies
	buy premium	BP	13	Premium plugins
	secure	S	8	Security mechanisms
	hosting bandwidth	HB	4	Fast bandwidth
	advanced testing techniques	ATT	5	advanced testing techniques

1) SECURE: in this context, emphasis is laid on enforcing security on already existing systems put in place with the help of frameworks. This is to counteract the effect of

hackers that have mastered the framework's vulnerabilities. Going in for additional security services like secured socket layer (SSL) certificates is an added advantage for system sustainability.

2) HOSTING BANDWIDTH: in this context, the bandwidth required to host a system should be high to enable the system to function properly and have a low response time. This also involve having a dedicated bandwidth for a given hosted system.
3) ADVANCED TESTING TECHNIQUES: this involves testing the system built with other testing frameworks like Jest framework use for JavaScript testing. Also, a secured version of the system should be sent out for a rewardable hackathon to reveal vulnerabilities.

Memo for Selective Coding

In this particularly study, six major themes emerge as the major reasons why software/I. S coded with software frameworks still fail despite several attempts to ensure software/I. S effectively and efficiently work without failing. These themes are; scalability issues (non-scalable systems), vulnerabilities (Software/I.S loopholes), update/upgrade (regular updates/upgrades), coding skills (inappropriate coding skills), null plugins (cost-free plugins), feasibility study (feasibility studies concerns) and hosting (hosting complications). As a response strategy to curb software/IS from failing, seven major themes emerge which are install updates (updates installation), support document (framework documentation), tech communities (tech communities), training (skillful training), system analysis (Preliminary studies), buy premium (premium plugins) and secure (security mechanisms).

Figure 1

A model of causes of software/I.S failure

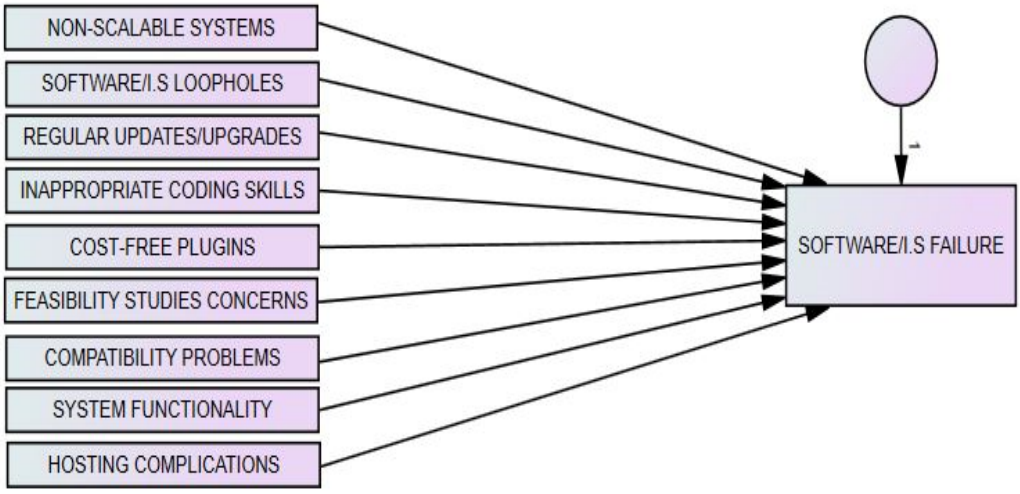

Figure 2
Statistical data on the causes of software/I.S failure

Figure 3
A model of counter measures to software/I.S failure

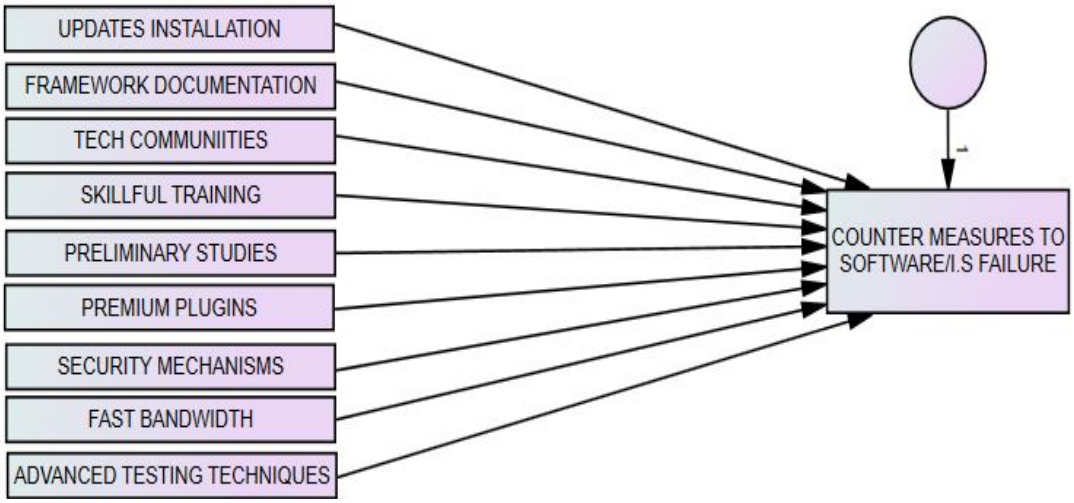

Figure 4
Statistical data on counter measures to software/I.S failure

DISCUSSION AND CONCLUSIONS

In the context of this study and more particularly the causes of software/I.S failure built with software frameworks, six major themes emerge as a major reasons why software/I.S coded with software framework fail. These themes are; vulnerabilities (Software/I.S loopholes), update/upgrade (regular updates/upgrades), coding skills (inappropriate coding skills), null plugins (cost-free plugins) and feasibility study (feasibility studies concerns). As a response strategy to the causes of failing system built with software frameworks, seven major themes emerge as counter measures against failing systems. These themes are; updates (updates installation), support document (framework documentation), tech communities (tech communities), training (skillful training), system analysis (Preliminary studies), buy premium (premium plugins) and secure (security mechanisms). With all of this, it is very certain that if the above-mentioned solutions or counter measures against failing I.S/software coded with software framework are put in place, then the failing rate of software coded with frameworks will greatly reduce.

SUGGESTIONS FOR FUTURE RESEARCH

As regards this study, in order to ensure long lasting security measures and counter measures put in place to overcome problems associated with failing I.S/software, researchers should embark on investigating how software framework source codes can be protected against unauthorized users with the help of A.I (artificial intelligent), modern encryption algorithms, and machine learning algorithm. This protection is aimed at detecting smelling codes of I.S/software coded with software frameworks and source codes of frameworks with hidden spyware or scripts that can cause vulnerabilities.

REFERENCES

Aulia, F., Yasin, I., Rahmanto, Y., & Trialih, R. (2022). Web-Based Petroleum Sales Accounting Information System. *Jurnal Ilmiah Sistem Informasi Akuntansi*, *2*(2), 77–81. https://doi.org/10.33365/jimasia.v2i2.2016

Avenier, M.-J., & Thomas, C. (2015). Finding one's way around various methodological guidelines for doing rigorous case studies: A comparison of four epistemological frameworks. *Systèmes d'information & Management*, *Volume 20*(1), 61–98. https://doi.org/10.3917/sim.151.0061

Ayuningtiyas, K., Suryanto, S., & Herachwati, N. (2023). *Smart Personality Development Model in Software Developer*. *3*(1).

Bianco, V. D., Myllarniemi, V., Komssi, M., & Raatikainen, M. (2014). The role of platform boundary resources in software ecosystems: A case study. *Proceedings - Working IEEE/IFIP Conference on Software Architecture 2014, WICSA 2014*, 11–20. https://doi.org/10.1109/WICSA.2014.41

Boell, S. K. (2015). *What is an Information System ? What is an Information System ? March*. https://doi.org/10.1109/HICSS.2015.587

Busetto, L., Wick, W., & Gumbinger, C. (2020). How to use and assess qualitative research methods. *Neurological Research and Practice*, *2*(1). https://doi.org/10.1186/s42466-020-00059-z

Dunwoodie, K., Macaulay, L., & Newman, A. (2022). *Qualitative interviewing in the field of work and organisational psychology : Benefits , challenges and guidelines for researchers and reviewers*. *June*. https://doi.org/10.1111/apps.12414

Dunwoodie, K., Macaulay, L., & Newman, A. (2023). Qualitative interviewing in the field of work and organisational psychology: Benefits, challenges and guidelines for researchers and reviewers. *Applied Psychology*, *72*(2), 863–889. https://doi.org/10.1111/apps.12414

Feinbube, L., & Polze, A. (2016). *The landscape of software failure cause models*. *June*.

Ihor, P. (2023). *ПОРІВНЯЛЬНИЙ АНАЛІЗ СИСТЕМ РОЗРОБКИ ПРОГРАМНОГО*. *2023*(317), 155–161.

Islam, R., Ahmed, A., Rahman, R., & Jany, R. (2022). A Survey of Software Metrics for Transitioning to an Updated Version of the Current Existing Framework or a New Framework. *Journal of Electrical Electronics Engineering*, *1*(1). https://doi.org/10.33140/jeee.01.01.04

Kaur, S., & Singh, S. (2023). Object Oriented Metrics Based Empirical Model for Predicting " Code Smells " in Open Source Software. *Journal of The Institution of Engineers (India): Series B*, *104*(1), 241–257. https://doi.org/10.1007/s40031-022-00833-4

Khalid, H., Shihab, E., Nagappan, M., & Hassan, A. (2014). What Do Mobile App Users Complain About? A Study on Free iOS Apps. *IEEE Software*, *PP*(99), 1–1. http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6762802

Khan, S. (2016). *Qualitative Research Method : Grounded Theory Qualitative Research Method : Grounded Theory*. *October 2014*. https://doi.org/10.5539/ijbm.v9n11p224

Khurshid, S., Shrivastava, A. K., & Iqbal, J. (2021). Effort based software reliability model

with fault reduction factor, change point and imperfect debugging. *International Journal of Information Technology (Singapore)*, *13*(1), 331–340. https://doi.org/10.1007/s41870-019-00286-x

Laudon C, K., & Laudon P, J. (2014). *Management Information Systems THIRTEENTH EDITION GLOBAL EDITION*. http://dinus.ac.id/repository/docs/ajar/Kenneth_C.Laudon,Jane_P_.Laudon_-_Management_Information_Sysrem_13th_Edition_.pdf

Mateen, A., Kausar, S., & Sattar, A. R. (2017). *A Software Reuse Approach and Its Effect On Software Quality, An Empirical Study for The Software Industry*. *July*, 670–678. http://arxiv.org/abs/1702.00125

Mealin, S., & Murphy-Hill, E. (2012). An exploratory study of blind software developers. *Proceedings of IEEE Symposium on Visual Languages and Human-Centric Computing, VL/HCC*, 71–74. https://doi.org/10.1109/VLHCC.2012.6344485

Mutanga, M. B. (2022). Secure Software Development Awareness: a Case Study of Undergraduate Developers. *Journal of Engineering Science*, *29*(2), 76–90. https://doi.org/10.52326/jes.utm.2022.29(2).08

Myllärniemi, V., Kujala, S., & Raatikainen, M. (2018). *Development as a journey : factors supporting the adoption and use of software frameworks*.

Nadu, T. (n.d.). *and Challenges in 21st Century Librarianship*.

Nascimento, C., & de Mello, R. (2022). Investigating the Perception of Success in Software Projects Among Developers from a Brazilian Software Company. *CIbSE 2022 - XXV Ibero-American Conference on Software Engineering*. https://doi.org/10.5753/cibse.2022.20971

Nazir, S. (2022). *Systematic analysis of software development in cloud computing perceptions*. *May*, 1–26. https://doi.org/10.1002/smr.2485

Nguyen, G., Dlugolinsky, S., Bobák, M., Tran, V., López García, Á., Heredia, I., Malík, P., & Hluchý, L. (2019). Machine Learning and Deep Learning frameworks and libraries for large-scale data mining: a survey. *Artificial Intelligence Review*, *52*(1), 77–124. https://doi.org/10.1007/s10462-018-09679-z

Noble, H., & Heale, R. (2019). Triangulation in research, with examples. *Evidence-Based Nursing*, *22*(3), 67–68. https://doi.org/10.1136/ebnurs-2019-103145

Overconfidence, M., Management, T., Pricing, T., & Exchange, I. S. (2023). *PURPOSIVE SAMPLING TECHNIQUE AND ORDINARY LEAST SQUARE ANALYSIS : INVESTIGATING THE RELATIONSHIP BETWEEN MANAGERIAL OVERCONFIDENCE , TRANSFER PRICING AND TAX MANAGEMENT IN INDONESIAN STOCK EXCHANGE- LISTED FIRMS Article history : Keywords : Managerial Ove*. 1–23.

Palacios Chavarro, S., Nespoli, P., Díaz-López, D., & Niño Roa, Y. (2023). On the Way to Automatic Exploitation of Vulnerabilities and Validation of Systems Security through Security Chaos Engineering. *Big Data and Cognitive Computing*, *7*(1), 1–24. https://doi.org/10.3390/bdcc7010001

Ratnawati, R., Kartikasari, E., Rustandi, B., Susanto, B., Setiawan, A., & Munastha, K. (2022). *A Web-Based Accounting Information System Application using CodeIngniter Framework: (A Case Study Approach)*. https://doi.org/10.1109/TSSA56819.2022.10063881

Reshi, J. A. (2017). *Software Maintenance : Role of metrics and Smells in object oriented*

softwares. *December 2016*.

Ribeiro, A., & Da Silva, A. R. (2012). Survey on cross-platforms and languages for mobile apps. *Proceedings - 2012 8th International Conference on the Quality of Information and Communications Technology, QUATIC 2012*, *September*, 255–260. https://doi.org/10.1109/QUATIC.2012.56

Semiawan, T., Riza, M., Hayati, H., Cahyo, D., & Lieharyani, U. (2021). *Analysis of the Effectiveness and Efficiency of Software Development Tools*. *207*(Issat), 32–39.

Sharma, T. (2019). *Detecting and managing code smells : research and practice Detecting and Managing Code Smells : Research and Practice*. *July*. https://doi.org/10.1145/3183440.3183460

Susanto, F. G. P., Fadlan, N. I. Y., & Haryani, P. (2023). Design of Web-Based Management Information System for Student Organizations in Kendal Regency Using Next.js Framework. *Compiler*, *12*(1), 9. https://doi.org/10.28989/compiler.v12i1.1616

Toman, Z. H., Toman, S. H., & Hazar, M. J. (2019a). an in-Depth Comparison of Software Frameworks for Developing Desktop Applications Using Web Technologies. *Xinan Jiaotong Daxue Xuebao/Journal of Southwest Jiaotong University*, *54*(4). https://doi.org/10.35741/issn.0258-2724.54.4.1

Toman, Z. H., Toman, S. H., & Hazar, M. J. (2019b). An In-Depth Comparison Of Software Frameworks For Developing Desktop Applications Using Web Technologies. *Journal of Southwest Jiaotong University*, *54*(4). https://doi.org/10.35741/issn.0258-2724.54.4.1

Wendler, R. (2012). The maturity of maturity model research: A systematic mapping study. *Information and Software Technology*, *54*(12), 1317–1339. https://doi.org/10.1016/j.infsof.2012.07.007

Xie, W., Li, Z., & Zhao, Y. (2013). *CoServices : A Web Service Based Middleware Framework for Interactive Cooperative Applications*. *37*, 507–513. https://doi.org/10.1109/ISDEA.2012.124

Ylönen, M., & Aven, T. (2023). A framework for understanding risk based on the concepts of ontology and epistemology. *Journal of Risk Research*, *26*(6), 581–593. https://doi.org/10.1080/13669877.2023.2194892

Zemmouchi-ghomari, L., Government, O., & Initiatives, D. (2021). *Basic Concepts of information systems*. *July*. https://doi.org/10.5772/intechopen.97644