Proceeding Paper

Software Reuse Practices among Malaysian Freelance Developers: A Conceptual Framework †

Mohd Akmal Faiz Osman *, Mohamad Noorman Masrek and Khalid Abdul Wahid

Faculty of Information Management, UiTM Cawangan Selangor Kampus Puncak Perdana, Seksyen U10, Shah Alam 40150, Malaysia
* Correspondence: akmalfaiz@uitm.edu.my

† Presented at the International Academic Symposium of Social Science 2022, Kota Bharu, Malaysia, 3 July 2022.

Abstract: Software reuse development practices have been proven to benefit software development in terms of a quicker time to market and quality. Although being aware of the benefits, software developers tend to overlook this method due to various reasons. Impacted by the advancement of technology, the software development scenario has been dramatically altered as the popularity of software development organizations hiring third party developers acutely rises. Freelance developers enable software development organizations to obtain high expertise in development specific areas on a project-by-project basis that gives development cost flexibility. However, information regarding freelance developer development practice is found to be limited in the related literature. By applying the systematic literature review approach, this study's main objective is to develop a conceptual framework to investigate software reuse practices among Malaysian freelance developers. The framework consists of individual characteristics, project characteristics, technological characteristics, software reuse development practices, quality, and achievement goals as constructs. The framework would be a basis for future empirical studies.

Keywords: software reuse; development practice; freelance developer

check for updates

Citation: Osman, M.A.F.; Masrek, M.N.; Wahid, K.A. Software Reuse Practices among Malaysian Freelance Developers: A Conceptual Framework. *Proceedings* **2022**, *82*, 30. https://doi.org/10.3390/proceedings2022082030

Academic Editor: Mohamad Rahimi Mohamad Rosman

Published: 13 September 2022

1. Introduction

Software reuse practices entails capitalizing on existing software and systems to create new products [1]. Instead of developing new software from scratch, developers use reusable software assets, such as components, information resources, source codes, conception, frameworks, and documentation to create new software systems [2]. The idea behind using reusable assets during the development of new software applications is to envisage an efficient way of development, rather than rewriting the modules that already exist. Software reuse practices can be applied in any of the software development phases such as requirement gathering, design, implement, testing, or maintenance. For instance, the development of software using reusable assets that have been tested rigorously by other developers will result in the testing phase to be shortened [3]. However, despite the benefits of adopting software reuse during software development, professional developers tend to overlook this method due to various reasons [4]. This has captured the attention of researchers, scholars, and the software development community [2,4] to investigate influencing factors of software reuse development practice. In general, among the prominent factors identified in literature are individual characteristics [2,4] and technological characteristics [5]. More thorough investigation is needed to support the informal view on the relationship by statistically presenting a thorough analysis of the relationship. Moreover, there are limited studies in the context of freelance software developers. Therefore, by applying the systematic literature review (SLR) approach by [6], the objective of this study is to propose a conceptual framework to investigate software reuse practices among Malaysian freelance software developers.

2. Freelance Software Developer

Freelance software developers are self-employed individuals who work on short term, project-based associations with hiring employers [7]. The term third-party developer explains the situation where software development organizations outsource software development tasks to a freelancer via freelancing websites or contacts. This is proven as [8] highlights the rapid increase in both the number of software development projects as well as tasks posted on online freelancing platforms, and this freelance industry is worth billions of dollars [9]. In 2018, it was reported that more than 9 million freelancers were registered in 110 freelance services websites, where the IT and programming section is one of the most important fields [9]. These freelance developers are available in various online marketplaces that provides flexibility for hiring organizations [10,11]. Many organizations have redefined their structures to incorporate freelance workers to development tasks [12]. Freelance developers possess highly professional skills in a specific area of software development, but there is no fixed monthly salary, insurance, pension, or any other benefits that are usually enjoyed by traditional developers [11]. The rapid advancement of IT technology has led to the software lifecycle being dramatically shortened where frequent updates were needed to cater to human software needs. This situation has prompted development organizations to find development alternatives by hiring these freelancers. To this end, freelance developers are required to develop the software of high quality more quickly.

3. Related Studies

Factors that influenced software reuse among 35 software developers were investigated by [5] and technological factors were found out to be the most significant predictors of software reuse, as reliability and availability of software assets are among the determinants. Similarly, findings by [13] found out that despite the awareness of software reuse benefits, developers tend to write new codes instead of reusing existing codes due to availability, accessibility and acceptability constraints of the reusable assets. In the research of [2] have presented similar findings with [5], in which technological factors and individual factors influenced intention to reuse software assets. In other research, software reuse practices have positive impacts on quality and efficiency [14]. In order to investigate the effect of software reuse development practice among freelance developer, [15] categorized two types of software reuse development practice, namely replicative usage and innovative usage. Although this study is still in its early stage, where the author only pilot tested the conceptual framework, this study is worth expanding into an empirical analysis as the study promotes innovative developer performance, which is crucial to freelance developers [7–10]. As shown in [2,5,13–15] studies, software reuse contributes to product quality and development quality, while individual characteristics and technological characteristics contribute to software reuse practice. However, the samples used from the studies was a traditional developer and not the freelance developer. Traditional developer works permanently in organization, in a team of other developers, and with considerable resources, such as funding, platform, and manpower [16]. Conversely, a freelance developer is self-employed and works on a project-by-project basis with limited resources [7].

4. Literature Review

4.1. Quality

Software that is developed using reused components results in better quality [1,17]. The reusable components used in the development have already been tested rigorously, with prior bug removal and defect corrections done. Findings from the SLR approach confirmed that product quality and development quality ranked highest in terms of the number of studies that have been published. To accommodate the quality of software development using freelancers, this study proposes two variables to measure the quality dimension, which are product quality and development quality.

4.1.1. Product Quality

Product quality is the extent of which a product or service meets or exceeds a client's expectations [15]. Software product quality includes having low defects, and being maintainable, portable, reliable, upgradable, and safe [14,15,17]. Direct relationship between the amount of reuse and level of software quality have been found in [17–19], while the quality of software is better when the software was developed using reusable components [20]. The consistency of the findings from the early 2000s until the 2020s have provided strong evidence of the effect of reuse towards software quality, and those freelance developers must be able to either replicate or innovate software reuse assets to develop quality software with efficiency [20].

4.1.2. Development Quality

Development quality is key performance indicator in software development [1,2,15]. Previous literature often pairs product quality and development quality as a measure of development success and quality [1,2,17,19,21]. Development quality captures the extent to which the software development process was managed with efficiency in terms of whether the software was completed on time and within budget [15]. The life span of software has been significantly shortened as human software needs are ever more complex, and more efficient development is needed. Therefore, it is assumed that software reuse practices would influence reduced software time to market and cost [1,2,17,19].

4.2. Achievement Goals

The difference between freelance developers and traditional developers are well noted in [11]. In terms of measuring freelance developer performance, achievement goals need to be included; these are defined as proficiency-relevant goals that an individual would make in relation to their competency [22]. Achievement goals are important in determining the career success and growth of a freelance developer, as they are self-employed individuals, compared to traditional developers who usually have a predetermined key performance indicator set by the attached organization. Hence, freelance developers would need to set their own achievement goals for career advancement and survival. Motivational factor of freelance developer was explored in [20]. The authors highlighted achievement goals that involves personal mastery and performance as significant predictors to motivation.

4.2.1. Satisfaction

Satisfaction is defined as a feeling of pleasure or displeasure that results from aggregating all the benefits that a developer hopes to receive from software reuse development practice [23]. Satisfaction sometimes is the best measure of success in terms of software development practices [24]. Satisfaction of software developers is closely related to motivation, where satisfied software developers were found to be more focused, committed, and hardworking [25]. Moreover, satisfaction of developers was found to influenced job productivity [26,27]. Instead of committing to repetitive tasks, reuse practices allow the expertise of freelance developers to be used more efficiently by focusing on developing higher quality software. Software reuse enables developers to complete pending projects quickly, sometimes ahead of schedule. With that said, therefore, satisfaction is critical to development practice success.

4.2.2. Personal Sense of Accomplishment

Personal sense of accomplishment is defined as the developer's feelings of self-esteem resulting from practicing software reuse [28]. In IS research, personal sense of accomplishment has been used to measure the success of certain practice effectiveness [29]. The performance of certain practices can only be considered successful if it gives a positive impact to the developer in enhancing their personal growth and productivity. Software reuse practices are considered to be development practices and their effectiveness can be measured through the developer's personal sense of accomplishment. A freelance developer's personal growth and productivity are of the utmost importance to their career, if not their survival.

4.3. Software Reuse Development Practices

In regard to the knowledge reuse theory, a conceptual framework has been developed by [15] to measure the impact of software reuse usage on software quality and development quality. According to [15], that software development jobs require a high level of innovation among developers to solve complex problems, and categorized the software reuse usage into two, namely replicative usage and innovative usage. Reusable assets, such as components, source codes, frameworks, and documentation, can be used accordingly with only slight modifications, or integrated and manipulated by combining them into novel components to develop a new software. In the freelance development world, with fierce competitors and an uncertain market, having the ability to replicate or innovate reusable software assets is considered a competitive advantage. The freelance developer could find the source and reusable assets and integrate them into a new software that satisfies the client's needs. Following and extending the study of [15], this study classifies software reuse practice into three forms: replicative usage, innovative usage, and extent of usage. Therefore, it is hypothesized that,

H1: *Software reuse development practices have a positive effect on Quality.*

H2: *Software reuse development practices have a positive effect on Achievement Goals.*

4.3.1. Replicative Usage

Replicative usage occurs when a developer reuses available modules to solve a common technical problem that involves a little adaptation or integration of the modules [15]. When there is a particular development problem that needs solving, the developer reuses previously created assets to solve the problem. As there is little adaptation or integration needed in this process, no novel results are generated. Replicative usage refers to best practices. This reusable module usually resides in the developer community with support from other developers in the form of development guides, templates, and tutorials.

4.3.2. Innovative Usage

Innovative usage occurs when the developer reuses available modules to explore a challenging problem that involves deliberate adaptation or integration of the modules [15]. Contrary to replicative usage, innovative usage refers to the reuse of software assets in novel ways such as combining and integrating existing software assets to develop a novel software asset that can be used again for a similar problem. Instead of using best practices that are available in the software community, the developer opts for the trial-and-error learning process, which leads to the creation of new assets. The developer starts with evaluating the reuse module to identify the intended function. By going through this process, developers are able to implement new functionalities into the software development process and attract more clients.

4.3.3. Extent of Use

Software reuse practice can be considered as a class of innovation [30]. Consequently, the diffusion of the innovation theory is adopted as a reference theory that implies to IS implementation and software reuse adoption. Extent of use captures the extent of which software reuse features and functionality are used in a complete and comprehensive manner in the development process [31]. Referring to [30], this study has modified the original extent of use of the CASE tool into extent of use of reusable software artefacts to suit the objective of the study. Previous studies have empirically shown that extent of use of software reuse has a positive impact on quality [14,17,31].

5. Individual Characteristics

Human factor was often found to contribute towards success or failure of software. Software developers differ from other developers in terms of their experience, expertise, and self-efficacy. From the literature, it is found that experience, expertise, and self-efficacy positively affect developer development practice [2,5].

5.1. Experience

Experience in the software engineering context can be defined as the extent of professional experience an individual possesses of being involved in systems and software development in years [32]. Experience enhances the capability and maturity of software developers, which is crucial in software reuse. Experience is listed as one of the variables to measure individual factor influence towards software reuse [5]. Similarly, experience was found to influenced software reuse practices adoption [4]. The accumulated experience of software development and reuse practice enables the developer to understand the value of the assets. Therefore, it is hypothesized that,

H3: *Experience positively influences software reuse development practice among freelance developers.*

5.2. Expertise

Expertise is defined as the degree of proficiency in a specific software development job. Recent studies have demonstrated that as organizational environments have evolved, work-related factors need to be revised through the perspective of professionals and the reality of modern software development companies [33], given, e.g., the emergence of developer expertise in front-end developers, back-end developers, full-stack developers, database architecture, and mobile apps developers. Domain knowledge and capability of developers are important in software reuse [5]. Moreover, expertise of a developer would determine innovative capabilities [34]. Therefore, it is hypothesized,

H4: *Expertise positively influences software reuse development practice.*

5.3. Self-Efficacy

In software development context, self-efficacy can be defined as individual self-judgment of their ability to use technology or innovation to accomplish a software development task [2]. Findings from [2] revealed that when developers are perceived to possess necessary skills to develop and reuse software assets, they are likely to practice software reuse. The practical reason of self-efficacy terms incorporated into software development is the assumption that an individual would most likely adopt a technology or innovation if they perceived themselves to have the ability to incorporate the practices in completing a task. Based on this assumption, it is hypothesized,

H5: *Self-efficacy positively influences software reuse development practice.*

6. Project Characteristics

Project characteristics concerns the characteristics of a software terms of the cost, duration and complexity [35]. Cost, duration, and complexity was found to influence the developer adoption of project management software to achieve higher productivity [35,36]. In the IS research context, project characteristics have a strong effect on system utilization [37]. Factors, such as increased number of modules in development, larger amount of information required, the needs of being able to track and reusing artefacts without developing a new one, will initiate the developer to embark upon software reuse practice.

6.1. Cost

Cost concerns with the amount of monetary budget allocated for development project [35]. Software development includes costs, such as purchase of hardware, license, components, templates, modules, knowledge, and training. While there is little known about the effect development cost on the adoption of software reuse, the cost of a software project could provide positive influence on the software reuse practice. When the development cost is high, developers thus need some innovation to reduce the cost. The total cost of the development project would encourage freelance developers to adopt software reuse practices. Therefore, it is hypothesized that,

H6: *Cost positively influences software reuse development practice.*

6.2. Duration

Duration concerns with the time consumed during development from start to market. Duration of software to be developed has influence the method used in the development [37]. Consider this example. Software project A took 12 months to complete that included 20,000 lines of code while software project B took 3 months to complete that consists of 5000 lines of code. Both projects very much differed in terms of duration to complete. It is interesting to see how freelance developers perceived software reuse regarding two different kinds of projects, as there is no information regarding this issue in the literature. Therefore, it is hypothesized that,

H7: *The duration of software projects influences software reuse development practice.*

6.3. Complexity

Complexity is the degree to which the software project is perceived difficult and challenging to complete [38]. Project complexity has been shown to determine the use of a variety of technologies, such as support systems [39]. From this proposition, it is possible that when freelance developers face a very complex task, they are more likely to perceive that software reuse will aid them to complete the task. Therefore, it is hypothesized that,

H8: *The complexity of software projects influences software reuse development practice.*

7. Technological Characteristics

In this study, technological characteristics pertain to technical characteristics of the reusable software assets. This construct is based on the diffusion of innovation theory by Rogers which posits the technological characteristic of innovation, such as compatibility, relative advantage, trialability, and result demonstration, that would influence the individual adoption of innovation. Studies have shown that these technological characteristics have influenced individuals to adopt specific innovation [31,40]. Although this theory proved to provide strong determinants in multiple contexts, this study would generalize and extend further into the context of freelance developers.

7.1. Compatibility

Compatibility is defined as the degree to which software reuse adoption is perceived as being consistent with the existing values, needs, and past experiences of the developer [40]. Freelance developers engaged in software development activities, such as software requirement gatherings, design specifications, source code writing, and software testing. Perhaps this kind of activity can be reused for new development [2,4]. For any developer, the compatibility and acceptance of development practice is important before they use it [5]. This study posits that,

H9: *Compatibility of reused assets influences software reuse development practices.*

7.2. Relative Advantage

Relative advantage is defined as the degree to which using software reuse is perceived as being better than developing from scratch [40]. Among important items in relative advantage is economic profitability, low initial cost, saved time, decrease in discomfort, and immediacy of rewards. Relative advantage has been found to be a consistent and strong predictor of individual adoption in various fields of IS studies such as software development practice [31,40]. However, in software reuse context, it is postulated by [31] that developers who believe that a new technology can enhance their task and job performance are more likely to use the new technology compared to those developers who do not hold this belief. Therefore, it is hypothesized that,

H10: *Relative advantage of reuse assets influences software reuse development practice.*

7.3. Trialability

Trialability is the degree to which an innovation may be experimented with on a limited basis [41]. Trialability plays a prominent role during the persuasion stage of the innovation process, as many reusable software artefacts usually are free to try and be experimented by developers. In the literature, trialability is often positively related to innovation adoption [31]. Trialability is an important trend towards adoption in the real world today, for instance, many software vendors usually provide users with a 30-day trial access. This study tends to widen the context to encompass freelance developers by positing,

H11: *Trialability of reuse assets influences software reuse development practice.*

7.4. Result Demonstration

Result demonstration is tangibility of the results of using innovation, consists of observability and communicability [41]. A clearly defined result from observation might result in adoption among developers. Freelance developers might find software reuse is convincing during development. Result demonstration proved to be an important determinant in IS adoption research [40]. Therefore,

H12: *Result demonstration influences software reuse development practices.*

8. Proposed Framework

Figure 1 shows the proposed framework that was developed based on previous relationships determined from previous literature. Experience of software developers was found to affect reuse development practice in the study of [2,4,5], where previous experience in systems development and adoption of software reuse would reduce the uncertainty of developers. Self-efficacy was found to influence reuse practices in [2,4], where the perceived competency of developers regarding reusable assets influenced their decision. Although project characteristics were not thoroughly investigated in software reuse literature, the relationship between software process improvement and adoption was discovered in the findings of [42]. Relationship between task complexity and knowledge reuse was discovered in findings of [39]. The same could be applied to software reuse practice as software reuse practice is considered as an improvement process using previous knowledge. Technological characteristics proved to be the strongest predictor of reuse practice, as showcased in the study of [30,31]. Software reuse practices were found to be strongly influenced both quality of developed software and development quality [1,2,14,17,19–21]. Satisfaction and a personal sense of accomplishment have been found to correlate with other development practices in the context of software development [26,27].

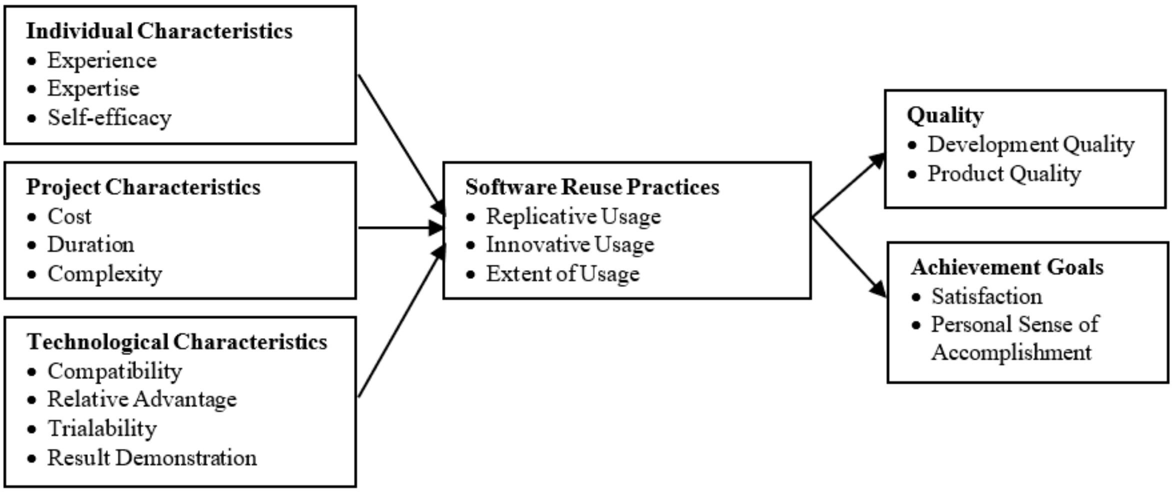

Figure 1. The proposed framework.

9. Discussion and Conclusions

This study proposed a conceptual framework to investigate the software reuse development practice among Malaysian freelance software developers, as there is scarce research in the literature discussing this topic. As the software crisis is well known, software development organizations have redefined their strategies to incorporate freelance developers to remedy the organization's lack of expertise in software development. However, comprehensive studies regarding freelance developer development practice are limited. Therefore, identifying important variables for further study would minimize this limitation. Individual characteristics of developers are varied. Some developers could be ten times more skilled than others. Software development projects differ from other projects, as some projects consist of 1000 lines of codes while other projects consist of 20,000 lines of codes. Information regarding development projects of freelance developers in the literature is unknown. By discovering the project characteristics of freelance developers, projects can then be profiled, determining the areas for improvement. Previous literature shows that many developers opt to develop software from scratch rather than use reusable assets due to their being incomplete, outdated, and lacking documentation. By investigating the technological characteristics of reusable assets used by freelancers, the templates, modules, functions, source codes, and documentation of reusable assets could perhaps be improved to encourage the increased usage of these assets.

Author Contributions: Conceptualization, M.N.M. and M.A.F.O.; validation, M.N.M.; formal analysis, M.A.F.O.; investigation, M.A.F.O.; resources, M.A.F.O.; data curation, M.A.F.O.; writing—original draft preparation, M.A.F.O.; writing—review and editing, M.N.M. and K.A.W.; visualization, M.A.F.O.; supervision, M.N.M. and K.A.W.; project administration, M.A.F.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research has not received any external funding.

Institutional Review Board Statement: Not Applicable.

Informed Consent Statement: Not Applicable.

Data Availability Statement: Not Applicable.

Acknowledgments: This research was partially supported by Universiti Teknologi MARA Kelantan Branch, Malaysia.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Barros, J.L.; Pinciroli, F.; Matalonga, S.; Martínez-Araujo, S. What software reuse benefits have been transferred to the industry? A systematic mapping studies. *Inf. Softw. Technol.* **2018**, *103*, 1–21. [CrossRef]
2. Mellarkod, V.; Appan, R.; Jones, D.R.; Sherif, K. A multi-level analysis of factors affecting software developers' intention to reuse software assets: An empirical investigation. *Inf. Manag.* **2007**, *44*, 613–625. [CrossRef]
3. Mohagheghi, P.; Conradi, R. An empirical study of software reuse defect-density. In Proceedings of the 26th International Conference on Software Engineering, Edinburgh, UK, 23–28 May 2004; pp. 282–291. [CrossRef]
4. Bakar, N.H.; Kasirun, Z.M. Exploring Practitioners Requirements Reuse Empirical Study. *Int. J. Softw. Eng. Technol.* **2014**, *1*, 33–42.
5. Tung, Y.H.; Chuang, C.J.; Shan, H.L. A framework of code reuse in open source software. In Proceedings of the 16th Asia-Pacific Networking Operating Management Symposium, Hsinchu, Taiwan, 17–19 September 2014.
6. Webster, J.; Watson, R.T. Analysing the past for prepare the future: Writing a review. *MIS Q.* **2002**, *26*, xiii–xxiii.
7. Gupta, V.; Fernandez-Crehuet, J.M.; Hanne, T. Freelancers in the development process: A Systematic Mapping Study. *Processes* **2020**, *8*, 1215. [CrossRef]
8. Sison, R.; Lavilles, R. Grounded theory of online development freelancing. In Proceedings of the International Conference on Information System, San Francisco, CA, USA, 13–16 December 2018; p. 26. Available online: https://aisel.aisnet.org/icis2018/behavior/Presentations/26 (accessed on 20 May 2022).
9. Haq, N.U.; Raja, A.A.; Nosheen, S.; Sajjad, M.F. Determinants of client satisfaction in web development projects from freelance marketplaces. *Int. J. Manag. Proj. Bus.* **2018**, *11*, 583–607. [CrossRef]
10. Mullins, N. Career-Related Attitudes, Competencies of Freelance Workers. Master's Thesis, University of Pretoria, Pretoria, South Africa, 2019.
11. Kazi, A.G.; Yusoff, R.; Khan, A.; Kazi, S. Freelancer: A Conceptual Review. *Sains Hum.* **2014**, *2*, 3. [CrossRef]
12. Hsieh, J.K.; Hsieh, Y.C. Internet-based freelance developers in app marketplaces. *Int. J. Inf. Manag.* **2013**, *33*, 308–317. [CrossRef]
13. Agresti, W.W. Software Reuse: Developers' Experiences and Perceptions. *J. Softw. Eng. Appl.* **2011**, *4*, 48–58. [CrossRef]
14. Ha, W.; Sun, H.; Xie, M. Reuse of embedded software in small and medium enterprises. In Proceedings of the 2012 IEEE International Conference on Management of Innovation & Technology (ICMIT), Bali, Indonesia, 11–13 June 2012; pp. 394–399.
15. Zhou, J. Application developer's innovation performance on mobile platforms-Investigating the effect of module reuse. In Proceedings of the 24th Pacific Asia Conference of Information System, Dubai, United Arab Emirates, 22–24 June 2020.
16. Sherif, K.; Vinze, A. Barriers to adoption of software reuse A qualitative study. *Inf. Manag.* **2003**, *41*, 159–175. [CrossRef]
17. Deniz, B.; Bilgen, S. Empirical study of software reuse and quality. *Lect. Notes Comput. Sci.* **2014**, *8583*, 508–523.
18. Frakes, W.B.; Succi, G. An industrial study of reuse, quality, and productivity. *J. Syst. Softw.* **2001**, *57*, 99–106. [CrossRef]
19. Rine, D.C.; Nada, N. An empirical study of a software reuse reference model. *Inf. Softw. Technol.* **2000**, *42*, 47–65. [CrossRef]
20. Slyngstad, P.; Gupta, A.; Landre, E. An empirical study on software reuse. *Int. Conf. Comput. Sci. Softw. Eng.* **2008**, *6*, 509–512.
21. Bauer, V.; Vetro, A. Comparing reuse practices in two large software-producing companies. *J. Syst. Softw.* **2016**, *117*, 545–582. [CrossRef]
22. Wulandari, A.; Qamara, T.; Bawazir, M. ELancing Motivation on Sribulancer. *J. World Conf.* **2019**, *1*, 193–202.
23. Masrek, M.N.; Jamaludin, A.; Mukhtar, S.A. Evaluating academic library portal effectiveness. *Libr. Rev.* **2010**, *59*, 198–212. [CrossRef]
24. Green, G.C.; Hevner, A.R. Successful doi: Guidance for software development organizations. *IEEE Softw* **2000**, *17*, 96–103.
25. Franca, C.; Da Silva, F.; Sharp, H. Motivation and Satisfaction of Software Engineers. *IEEE Trans. Softw. Eng.* **2020**, *46*, 118–140. [CrossRef]
26. Mannaro, K.; Melis, M.; Marchesi, M. Empirical analysis on satisfaction of IT employees. *Lect. Notes Comput. Sci.* **2004**, 166–174. [CrossRef]
27. Graziotin, D. Happiness and the Productivity of Software Engineers. *Rethinking Prod. in Soft. Eng.* **2019**, 109–124. [CrossRef]
28. Samadi, I.; Masrek, M.N.; Yatin, S.F. The effect of individual characteristics and digital library characteristics on digital library effectiveness: A survey at university of Tehran. *World Appl. Sci. J.* **2014**, *30*, 214–220.
29. Masrek, M.N.; Karim, N.S.A.; Hussein, R. Measuring corp. *Intranet effectiveness. Info. Manag. Comput. Secur.* **2006**, 89–112. [CrossRef]
30. Iivari, J. Why are case tools not used? *Commun. ACM* **1996**, *39*, 94–103. [CrossRef]
31. Kishore, R.; McLean, E.R. Reconceptualizing innovation compatibility as organizational alignment in secondary IT adoption contexts: An investigation of software reuse infusion. *IEEE Trans. Eng. Manag.* **2007**, *54*, 756–775. [CrossRef]
32. Ahmad, M.A.; Ubaidullah, N.H.; Lakulu, M. Current Practices in Monitoring Software Development Process in Malaysia. *World Comput. Sci. Inf. Technol. J.* **2014**, *4*, 62–67.
33. Magalhaes, C.; Santos, R. The Role of Job Specialization in the Software Industry. In Proceedings of the International Conference on Information Technology & Systems, Libertad City, Ecuador, 4–6 February 2021.
34. Fulk, H.; Nagy, D. Expertise and Information Technologies: A Multidisciplinary Review. In Proceedings of the Twenty-Sixth Americas Conference on Information Systems, Salt Lake City, UT, USA, 15–17 August 2020.
35. Pellerin, R.; Perrier, N.; Guillot, X.; Léger, P.M. Project characteristics, project management software utilization and project performance: An impact analysis based on real project data. *Int. J. Inf. Syst. Proj. Manag.* **2013**, *1*, 5–26. [CrossRef]

36. Ottow, J. The Individual Adoption of New IT System within Organizations. Master's Thesis, Tilburg University, Tilburg, The Netherlands, 2016.
37. Xu, J.; Quaddus, M. Examining a model of KM systems adoption and diffusion: A PLS approach. *Knowl. Based Syst.* **2012**, *27*, 18–28. [CrossRef]
38. Morgeson, F.; Humphrey, S. Job design: Toward integrative conceptualization. In *Research in Personnel and Human Resources Management*; Emerald Group Publishing: Bingley, UK, 2008; Volume 27, pp. 39–91. [CrossRef]
39. Iyer, G.; Ravindran, S. Do task complexity and knowledge recency affect knowledge reuse? Implications for knowledge management efforts. In Proceedings of the 13th Americas Conference of Information Systems, Keystone, CO, USA, 10–12 August 2007; p. 493.
40. Kügler, M.; Smolnik, S.; Raeth, P. Why don't you use it? Assessing the determinants of enterprise software usage: A conceptual model integrating innovation diffusion and social capital theories. *Int. Conf. Inf. Syst.* **2012**, *5*, 3672.
41. Rogers, E.M. Innovation diffusion at the implementation stage of a construction project: A case study of information communication technology. *Constr. Manag. Econ.* **2006**, *24*, 321–332.
42. Mangalaraj, G.; Mahapatra, R.; Nerur, S. Acceptance of SPI: The case of extreme programming. *Eur. J. Inf. Syst.* **2009**, *18*, 344–354. [CrossRef]