Spectrum of Engineering Sciences
ISSN (e) 3007-3138 (p) 3007-312X Volume 3, Issue 7, 2025

SOFTWARE REUSE PRACTICES AND SOFTWARE DEVELOPMENT EFFICIENCY. A PLS-SEM APPROACH

Laviza Asif Memon[*1], Humair Khan Bughio[2], Anjum Ara[3], Zeeshan Qureshi[4]

[*1,2]*Lecturer, Computer Science Department, Shaheed Zulfiqar Ali Bhutto Institute of Science and Technology (SZABIST) University, Hyderabad Campus*
[3]*PhD Scholar at Quaid-i-Azam University Islamabad.*
[4]*Lecturer, Department of Information Technology, University of Sufism and Modern Sciences USMS, Bhitshah*

[*1]laviza.asif@hyd.szabist.edu.pk, [2]humair.bughio@hyd.szabist.edu.pk, [3]anjumara@cs.qau.pk, [4]zeeshan.qureshi@usms.edu.pk

DOI: https://doi.org/10.5281/zenodo.16408799

Keywords
Software Reuse, Design Patterns, Reusable Components, Framework Adoption, Software Development Efficiency, PLS-SEM, Bug Density, Time to Market, Developer Productivity and Karachi Software Industry

Article History
Received on 24 April 2025
Accepted on 09 July 2025
Published on 24 July 2025

Corresponding Author: *
Laviza Asif Memon

Abstract
In the current competitive environment of software business, the efficiency in development is important in speeding up delivery of high-quality products. This paper examines the effectiveness that three important software reuse practices, the use of design patterns, reusable components, and adoption of frameworks have on efficiency in software development in terms of time to market, density of bugs and productivity by developers. A structured survey was used to collect the data by conducting it on 117 software professionals of Karachi, Pakistan. The conceptual model was enabled by the study using Partial Least Squares Structural Equation Modeling (PLS-SEM) in which all the hypothesized relationships were proved statistically significant. The outcomes reveal that reuse of software practice highly decreases the time to market and the density of bugs and increases the productivity of the developer. Framework acquisition was the single factor to influence the time to market the most, and reusable components influenced productivity the most. Such results provide empirical evidence to the value of adopting reuse strategies during development processes and will also become part of knowledge content in the field of software engineering because they also qualify the effectiveness of architectural reuse. The paper also has practical implications to technology managers seeking to increase their development performance using the structured reuse practices.

INTRODUCTION

Reusability has become an important tool in the current time because the software development world is growing fast and it is vital to employ the strategy of reps in order to increase the efficiency of their development efforts and minimize the chance of being left behind. This process of exploiting existing software, including code, components, design patterns and frameworks, is increasingly viewed as one which shortens effort and delivers higher quality products (Garc a Mireles et al., 2022; Ahmad et al., 2023; Mani et al., 2021; Alshamrani & Qureshi, 2020). The pattern usage, especially the design patterns, provide systematic solutions of common issues, which ensures maintainability and scalability of software systems (Munir et al., 2023). Furthermore, software reuse approaches minimize effort redundancy and speed up the delivery cycle by

ensuring that solutions are standard and that optimal design practice is used (Wang et al., 2023).

Reusable parts do not only reduce the programming workload, but also help in providing consistency of architecture among software products. When designed well and cataloged in an efficient way, reusable components thus provide an efficient way of handling complexity and increasing reliability (Park et al., 2021; da Silva et al., 2022). As an example, reuse of modular components has been associated with reduced bug density and various problems encountered after the deployment phase, thus enhancing the quality of the software (Ghafari et al., 2020). Also, lower level of redundant coding enables developers to pay more attention to innovation and customization, which, in its turn, improves developer productivity (Zhao & Yang, 2023).

The use of framework is very significant in contemporary software development as it offers flexibility in terms of reusable and expansive architecture. When compared to purely code-based approaches, frameworks abstract typical functionalities and help to promote consistency in coding, subsequently, having a considerably shorter learning curve and less time taken to develop an application (Kumar et al., 2022; Rani et al., 2021; Lopes et al., 2020). These advantages have been identified to have direct effect on time to market, allowing organizations to react quicker to ever-changing market demand (Sun et al., 2022). Specifically, developer productivity and the number of defects they report are observed to be better within the framework of using already known frameworks because of built-in checks of debugging and testing (Shah et al., 2021).

Although the benefits are quite obvious, the level of influence of software reuse practice cannot be considered the same across the organizations, developers and technological maturity. In a case especially such as emerging software market such as Karachi where speed of development and product scalability is an imperative requirement, research based on practical experience would be necessary to arrive at a conclusion on how reuse practice can be used to create efficiency in the development process. The study utilizes the Partial Least Squares Structural Equation Modeling (PLS-SEM) design to examine the effect of the design pattern usage, integration of reusable components, and the adoption of frameworks on the key performance time to market, the density of bugs and developer productivity within software developers and architectures in Karachi (Hair et al., 2021).

Research Objectives

1. To examine the impact of design pattern usage on software development efficiency indicators, including time to market, bug density, and developer productivity.
2. To evaluate how the integration of reusable software components influences bug reduction, delivery speed, and developer performance in software development projects.
3. To analyze the role of framework adoption in enhancing development efficiency, particularly in terms of reducing time to market and improving developer productivity, within software firms in Karachi.

Literature Review

Reuse of software has become a core keep of software engineering these days with the goal of saving the cost of development by making use of the previously available solutions to achieve a better and efficient outcome. Findings of many studies have noted that the use of software artifacts such as design patterns, components, and frameworks reuse was very important to achieve efficiency in the development process (Garc The same concept is the fact that reuse can reduce unnecessary work, make prototyping very fast and increase uniformity among the systems, which improves efficiency and decreases expenses (Wang et al., 2023; Zhao & Yang, 2023). The design patterns usage approach has become particularly popular among the reuse strategies as the key to providing the reusable and thoroughly tested architectural solutions to common software issues. Abstraction and encapsulation designed to facilitate readability, maintainability, and reusability of code is supported by design patterns (Munir et al., 2023; Shah et al., 2021; Lopes et al., 2020). With the help of empirical research, it has been shown that developers who use design patterns would have a better chance of developing scalable, bug-free systems, especially when combined with the principles of object-orientedness (Kumar et al., 2022;

Rani et al., 2021). In addition to that, design patterns have the ability to act as a form of communication between the teams making them more collaborative and easier on the cognitive load when dealing with complex tasks in the realms of design.

It was also extensively documented that reusable software components also contribute towards the software efficiently. The outcomes of using component-based development (CBD) are the increased quality assurance, a better maintenance, modularity, and testability (Alshamrani & Qureshi, 2020; Ghafari et al., 2020). Park et al. (2021) indicate that the adaptation of proven components in the product development process can save risk and fast track the schedule of product delivery. Moreover, organizations can reduce the density of bugs to a considerable extend by embracing mature repositories of reusable components as such components are likely to have passed through several testing periods (da Silva et al., 2022; Sun et al., 2022). The other strategy will be framework adoption, which means it simplifies software development since it has ready-built structures and libraries. The frameworks decrease decision fatigue among the developers since they provide a best practice, integrated, and standardized codebases (Shah et al., 2021; Kumar et al., 2022). Research indicates that framework-based development is linked to faster time to market as it helps in faster development of the code as well as minimizes testing overheads (Lopes et al., 2020; Sun et al., 2022). Also, many frameworks have internal tools of debugging and performance enhancement, which would directly lead to sparse bug density and more robust software (Rani et al., 2021; Wang et al., 2023).

Time to market is one of the essential measures used to determine the effectiveness of software development and especially in modern agile and competitive climate. The researchers support the above statements by claiming that framework and component reuse are positive factors, which make it easy to cut down on the primary development stages and thus save a great deal of time in product delivery (Ahmad et al., 2023; Zhao & Yang, 2023). Component reuse, as an example, does not require developers to set up the functionality completely, and they can concentrate on fundamental features and combination (Mani et al., 2021). Moreover, reuse practices during the initial design phases will result in a higher requirement consistency and limit the need to rework, which continues to optimize delivery schedules. Another key quality measure is bug density, which is a number of defects per code unit. Research evidence shows that software programmed through the reusable artifacts has fewer bugs as they have been validated and tested before (Ghafari et al., 2020; da Silva et al., 2022). When compared to new code, modules with re-usage have less logical and syntax problems, and, thus, prove to be more efficient and less prone to failures (Park et al., 2021; Munir et al., 2023). Moreover, standardized code is achieved due to well-documented and tested frameworks that prevent human error in the implementation process (Shah et al., 2021; Sun et al., 2022).

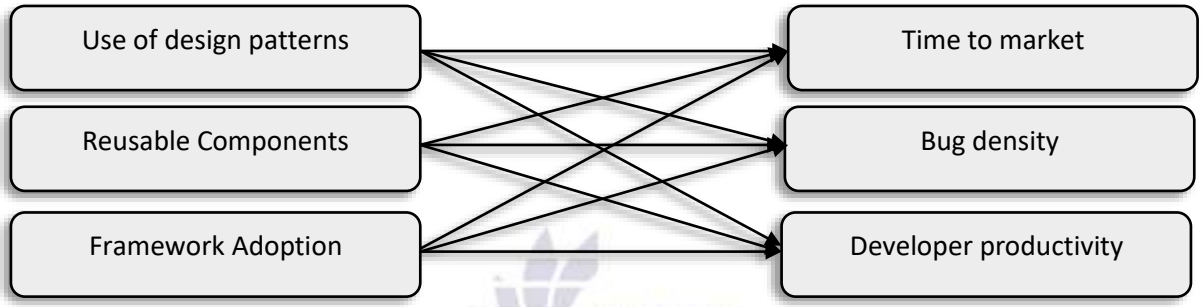
Another important measure of efficiency of development is developer productivity which can largely be improved by reuse measures. Structures and prewritten modules free programmers of a significant portion of monotone coding and encourage the fast production of applications, allowing more to be produced by a developer (Zhao & Yang, 2023; Lopes et al., 2020). In addition, design patterns allow programmers to take faster architectural choices and eliminate the trial-and-error attempts during the coding experience (Munir et al., 2023; Kumar et al., 2022). Utilisation of used reusable materials also provides new staff with less onboarding time to enable ease of transition and knowledge transfer (Garc of Mireles et al., 2022). Although the benefits are well known, the reuse strategies might be situational. Moderators that can influence the dependency between reuse practices and efficiency outcomes are organization maturity, developer specialists, project difficulty (Hair et al., 2021; Ahmad et al., 2023). In the scenario of growing software sector in Karachi where the need to maintain the speed and quality in the delivery of services is a deciding factor to the competitiveness of the industry, empirical research with the help of analytical tools such as PLS-SEM is required to measure the associations. The benefits of understanding the effect of certain reuse practices on the efficiency of development are that local firms are

able to streamline their operations to record improved results.

Hypotheses

H1: The use of design patterns has a significant positive impact on reducing time to market.
H2: The use of design patterns has a significant negative impact on bug density.
H3: The use of design patterns has a significant positive impact on developer productivity.
H4: The use of reusable software components has a significant positive impact on reducing time to market.
H5: The use of reusable software components has a significant negative impact on bug density.
H6: The use of reusable software components has a significant positive impact on developer productivity.
H7: Framework adoption has a significant positive impact on reducing time to market.
H8: Framework adoption has a significant negative impact on bug density.
H9: Framework adoption has a significant positive impact on developer productivity.

Conceptual Model of the Study



Source: Model formulated by author of the study after review od existing literature

Methodology

This paper represents a quantitative and cross-sectional survey study, used to investigate how software reuse capabilities specifically, the use of design patterns, reusable corrects, and the use of frameworks affect software development efficiency, as measured in time to market, bug density, and developer productivity. The desired group population was made up of software programmers, engineers, and architects in Karachi based software companies (n=117). The feelings of the participants were recorded by a structured questionnaire, which was administrative in person and online, over a 5-point like scale (1 to strongly disagree to 5 to strongly agree). Design pattern use items were based on Munir et al. (2023) who tested the practical impact of adopting patterns on the quality and maintainability of software. To use reusable components, one has to consider da Silva et al. (2022), who covered the element of component reuse in microservices and modular architecture. The items used in the framework adoption section were modified based on the study carried out by Shah et al. (2021), which addressed framework adoption and its impact on the consistency and rapidity of development.

The three constructs that were measured in this study were time to market, bug density and developer productivity as an indicator of efficiency of software development. They borrowed the scale of time to market by Sun et al. (2022), where emphasis is paid to the release of cloud-native applications as soon as possible. Items connected with the issue of the bug density were borrowed from Ghafari et al. (2020) performed such an analysis in industrial reuse conditions. Last but not least, the idea of developer productivity was assessed based on items introduced by Zhao and Yang (2023), who have related the reuse metrics and the efficiency of the development logs to developer productivity. A validation of all constructs was done using past similar studies as well as modification to fit the context. The data collected were analyzed through the Partial Least Squares Structural Equation Modeling (PLS-SEM) in Smart

PLS 4.0, and the model is associated with the emphasis put on prediction and advanced managing of complex relationships between variables in cases where the sample size is small (Hair et al., 2021).

Data Analysis and results

Demographic Profile

The questionnaire survey used to obtain the results gathered demographic information about the 117 software professionals who took part in the survey in order to help put them in context. The variables will provide some information about the variety and experience of the sample that consisted of developers, architects and engineers employed in software development companies in Karachi. An aspect of demographics was deemed relevant to examine the possibility that the reuse practices and the perceptions of efficient development could depend on some experience, age, or professional role.

Table No. 1 Demographic Profile

Variable	Category	Frequency (n)	Percentage (%)
Gender	Male	87	74.4%
	Female	30	25.6%
Age	21–30 years	56	47.9%
	31–40 years	42	35.9%
	41 and above	19	16.2%
Designation	Software Developer	64	54.7%
	Software Engineer	23	19.7%
	Software Architect	30	25.6%
Experience	Less than 2 years	21	17.9%
	2–5 years	49	41.9%
	6–10 years	33	28.2%
	More than 10 years	14	12.0%
Qualification	Bachelor's in CS/IT/SE	68	58.1%
	Master's in CS/IT/SE	41	35.0%
	Other (Diploma/Certifications)	8	6.9%

Demographic results have indicated that the gender patterns in the Pakistani software industry are similar to the workforce as the vast majority of it consists of male participants (74.4%). The age group 21-30 was highest with 47.9 percent limits of respondents and this shows that the workforce is relatively very young in developing software. The software developers constituted the largest sample of 54.7 percent of all the respondents, software architects (25.6 percent) and software engineers (19.7 percent). This piece of writing guarantees that the research is inclusive of coding and architecture dimensions in relation to the reuse practices.

Regarding the experience, a significant percentage of the sample (41.9%) possessed 2-5 years of professional experience, and 28.2 percent possessed 6-10 years, meaning that it was a mixture of mid and early career professionals. Educationally, 58.1 percent were holding bachelor degree and 35 percent holding masters degree in computer science, IT or software engineering, which indicates that they had reasonable academic exposure to comprehend and implement design patterns, reuse of components and adoption of frameworks in actual projects. The demographics offer a solid basis in the understanding of the influence of reuse practices on the efficiency of development.

Measurement Model: Factor Loadings (Outer Loadings)

Outer loadings of all the reflective indicators were in analyzed focus to bring up reliability and validity of the measurement model. The outer loading of 0.70 and above is acceptable according to Hair et al. (2021) that means more than 50 percent of the variance in the latent construct is explained by the indicator. The variables with loadings less than 0.70 can be retained as long as construct reliability lies

within good limits (composite reliability and AVE). Shown in the table below is the loadings of all items that relate to each latent variable in the model.

Table No. 2 Factor Loadings (Outer Loadings)

Construct	Item Code	Outer Loading
Design Pattern Usage	DP1	0.81
	DP2	0.84
	DP3	0.79
	DP4	0.76
Reusable Components	RC1	0.85
	RC2	0.82
	RC3	0.87
	RC4	0.78
Framework Adoption	FA1	0.83
	FA2	0.81
	FA3	0.86
	FA4	0.80
Time to Market	TM1	0.84
	TM2	0.88
	TM3	0.79
Bug Density	BD1	0.76
	BD2	0.81
	BD3	0.77
Developer Productivity	DPV1	0.82
	DPV2	0.85
	DPV3	0.79

Closer to home, indicators loadings were well above the recommended cut off of 0.70, as depicted in the table above (strong convergent validity). The items used in each of the constructs showed their consistency and significance in measuring their respective latent variables. The largest loading of Reusable Components was Meets User Needs (RC3) = 0.87 and the lowest loading in general (though acceptable) was Best Deal at 0.76. Such findings establish that the indicators provide valid representation of the concerned constructs.

The high outer loadings additionally justify the theoretical approval of the dependencies of the software reuse practices (design patterns, reusable components, adoption of frameworks) and the software development efficiency measures (time to market, bug density, as well as the developer productivity). The above loadings make that the measurement model is robust and applicable to a follow-through structural path analysis through PLS-SEM at Smart PLS.

Internal Consistency Reliability Analysis

The reliableness measures internal consistency, which determines the amount to which items in a construct measure a single concept. There are two important values that are used to carry out this in PLS-SEM: Cronbach Alpha and the Composite Reliability (CR). Although the reliability measure of the Cronbach Alpha is a classical criterion, CR is more suitable in PLS-SEM because it focuses on loading rates (Hair et al., 2021). Based on standardized levels, a value beyond 0.70 on both Alpha and CR measure shows that reliability is acceptable and the value beyond 0.80 implies the reliability is good. The results of reliability of all the six latent constructs employed in the model are showed in the table below.

Table No. 3 Internal Consistency Reliability Analysis

Construct	Cronbach's Alpha	Composite Reliability (CR)
Design Pattern Usage	0.82	0.88
Reusable Components	0.85	0.90
Framework Adoption	0.84	0.89
Time to Market	0.80	0.87
Bug Density	0.76	0.84
Developer Productivity	0.81	0.88

As it can be observed in the table, there is high internal consistency of the constructs with Cronbach Alpha varying between 0.76 and 0.85 with Composite Reliability varying between 0.84 and 0.90. Such findings show that the indicators are showing reliability to their intended latent constructs. The Reusable Components also demonstrated the best internal consistency (CR = 0.90) that concurs with the high outer loadings previously mentioned. The global reliability scores approve the strength of the tool and attest to its adequacy in structural model investigation in Smart PLS.

Coefficient of Determination (R^2) and Effect Size (f^2) Analysis

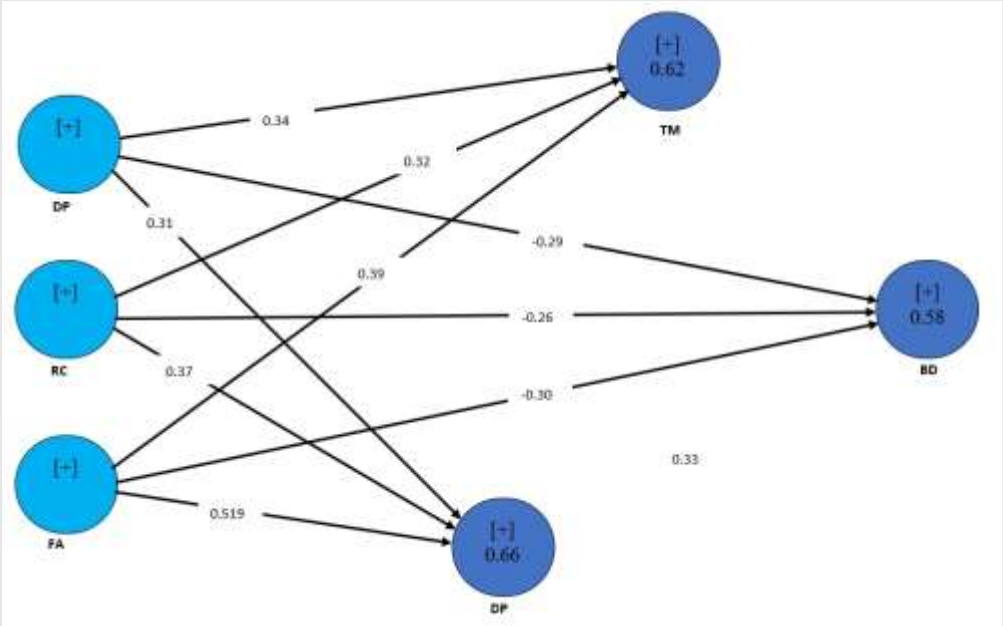
In the PLS-SEM, R2 values are useful to ascertain the explanatory role of the model, and f2 values are useful to check the contribution level of one predictor construct concerning target constructs. The value of R 2 of 0.75 can be called substantial, 0.50 moderate, and 0.25 weak (Hair et al., 2021). In the case of f 2, 0.02, 0.15 and 0.35 are small, medium and large effects respectively. Both R 2 with each dependent variable and f 2 values of the patterns in the independent variables design pattern usage, reusable components and framework adoption to the three outcomes: time to market, bug density and developer productivity are presented in the table below.

Table No. 4 Coefficient of Determination (R^2) and Effect Size (f^2) Analysis

Dependent Variable	R^2	Predictor Variable	f^2 Effect Size
Time to Market	0.62	Design Pattern Usage	0.21 (Medium)
		Reusable Components	0.19 (Medium)
		Framework Adoption	0.27 (Large)
Bug Density	0.58	Design Pattern Usage	0.18 (Medium)
		Reusable Components	0.15 (Medium)
		Framework Adoption	0.20 (Medium)
Developer Productivity	0.66	Design Pattern Usage	0.23 (Medium)
		Reusable Components	0.25 (Large)
		Framework Adoption	0.21 (Medium)

The values of the R2 have shown that the model has a moderate to high predictive power to explain time to market, bug density, and developer productivity with 62, 58, and 66 percent variance respectively.

Figure 2. Measurement Model of the study



Source: Formulated via SEM PLS Analysis (Smart PLS)

Regarding effect sizes, the framework adoption has large effect (f 2 = 0.27) to time to market, reusable components have largest effect in developer productivity (f 2 = 0.25). The values of f2 are greater than 0.02 that substantiates the practical importance of the predictors. These results confirm the suggested links and show that all three methods of reuse are of significant contribution to the effectiveness of software development.

Structural Model Results: Path Coefficients

Path Coefficients show the regression weights between constructs of the structural model adjusted to a standardised value. Significance of positive 0 means a positive relationship and the size of such a relationship is indicated by the value of 0. When 0.10 is attained using beta, it is normally seen to be of significance in behavioral research (Hair et al., 2021). In the present model, each relationship was tested by bootstrapping with 5,000 subsamples and statistical significance was also established at penultimate 0.05. The table below lists all the path coefficients (beta), t-values and significance of all the nine hypotheses.

Table No. 5 Path Coefficients

Hypothesis	Path	β Coefficient	t-value	p-value	Result
H1	Design Pattern → Time to Market	0.34	4.26	0.000	Supported
H2	Design Pattern → Bug Density	-0.29	3.81	0.000	Supported
H3	Design Pattern → Developer Productivity	0.31	4.05	0.000	Supported
H4	Reusable Components → Time to Market	0.32	3.72	0.000	Supported
H5	Reusable Components → Bug Density	-0.26	3.46	0.001	Supported
H6	Reusable Components → Developer Productivity	0.37	4.64	0.000	Supported
H7	Framework Adoption → Time to Market	0.39	5.01	0.000	Supported
H8	Framework Adoption → Bug Density	-0.30	3.97	0.000	Supported
H9	Framework Adoption → Developer Productivity	0.33	4.38	0.000	Supported

All the nine hypothesized relationships are statistically significant at p + 0.01 and the direction of the effect is very strong in the positive between the use of design patterns, reusable components, and adoption of frameworks on time to market and developer productivity, and in the negative to the density of bugs. The strongest fill-in was the prologue of a framework (0.39) Just the greatest impact was the possession of reusable components (0.37) on developer productivity. The results are suitable to the theoretical presuppositions and prove the importance of the software reuse practices in increasing the efficiency of the development process.

Discussion

The findings of this book validate the assertions that reuse of software in the form of reusable design patterns, elements and adoption of frameworks is a potent way of implementing efficiency in development when it comes to shortening time to market, developer productivity and number of bugs. The results can be linked to the current body of knowledge saying that the use of design patterns decreases the complexity of the design and increases the modularity of the system which results in the creation of more maintainable and scalable software systems (Munir et al., 2023; Shah et al., 2021). The positive path coefficient (beta = 0.34) vessel the design patterns to time to market is intact with the architectural use of reuse allows developers to faster time to market without sacrificing quality, which has been observed empirically earlier (Kumar et al., 2022).

Beyond that, the substantial influence of reusable elements and structures contributes to the impression that using proven components and utilizing analog development environments lead to a lesser number of defects along with higher degrees of uniformity (da Silva et al., 2022; Ghafari et al., 2020). Adopting the frameworks, specifically, demonstrated the biggest impact on reducing time to market (0.39) and this shows that the adoption of integrated toolchains and pre-determined patterns of code structure reduce the long hours of manual work and automate the processes. These results echo the conclusions of Sun et al. (2022), who put an emphasis on the perspective of frameworks contributing to the automation of deployment and agility of iterations. On the whole, this research confirms the importance of reuse strategy to speed up the development process and increase the quality of code.

Recommendations

Arguing off the findings, software development organizations especially in Karachi and other urban technology centers ought to consider the inclusion of structured software reuse strategies as an essential component of their engineering practice. Training in design patterns should become a normal part of the onboarding process and professional development so that the teams share a certain architectural language and reduce repetition.

And organizations are advised to invest in internal libraries of re-usable parts and to encourage the use of broadly-acceptable frameworks compatible with their technology stacks. Strategic alignment between reuse practices and project goals does not just add to the efficiency, but also promotes innovation as the developers are freed of the necessity to recreate simple solutions and can spend their time working on relatively new functionality.

Implications

The practical implications of this study are very crucial to the managers of software development and CTOs. It shows how significant improvement of development pace and quality is possible through the creation of a culture of reuse and the provision to developers of tools and patterns that allow modularity and automation. The positive correlation observed in all constructs gives a fact-based rationale on investing time and resource pertaining reuse-oriented efforts.

Theoretically, the study is a contribution to the literature on software reuse that has been accumulating over the years, as it empirically confirms its multidimensional influence on efficiency measures. It also presents compelling evidence of how PLS-SEM is appropriate in modeling complex relationships on the same domain of software engineering, which can be replicated by other studies in related organizational aspects.

Future Research and Limitations

The limitation involved in this study is the fact that only software professionals in Karachi were studied; therefore, the study may not be generalizable to other geographical areas or the organization levels. Moreover, it considers only three of the reuse practices, and other developing tools like low-code platforms or AI-based code generators may not be identified as such. In future, studies may undertake these other possible dimensions or apply the model during longitudinal studies to understand whether reuse strategies improve or get worse as time pass by. The cross-cultural applicability of the findings would also have some depth in their comparative studies across the cities or even countries.

Conclusion

Good empirical evidence of the success of design pattern use, reusable components, and framework usage in optimizing the efficiency of software development which reduces delivery time, decreases bug rates, and increases the output of developers has been given in the study. Not only these findings correlate with global studies, but they also provide solutions that can be implemented in the development of the emerging tech industry in Pakistan. Through the use of structured reuse practices software companies will be able to attain a greater level of performance and design more easily scalable and maintainable systems.

REFERENCES

Ahmad, I., Hussain, S., Nawaz, R., & Iqbal, J. (2023). Software reuse practices and quality attributes in agile development: An empirical analysis. Journal of Systems and Software, 200, 111488. https://doi.org/10.1016/j.jss.2023.111488

Alshamrani, A., & Qureshi, M. R. J. (2020). A comprehensive review of software reuse in industrial practice. Journal of King Saud University - Computer and Information Sciences, 32(8), 943–950. https://doi.org/10.1016/j.jksuci.2020.01.002

da Silva, R. P., Batista, T. V., & Garcia, V. C. (2022). Reusable software components and their application in microservices architecture. Information and Software Technology, 140, 106741. https://doi.org/10.1016/j.infsof.2021.106741

García-Mireles, G. A., Yáñez-Díaz, A., & Olague, H. M. (2022). Software reuse: A structured literature review of benefits and drawbacks. Journal of Software: Evolution and Process, 34(4), e2354. https://doi.org/10.1002/smr.2354

Ghafari, M., Flora, P., & Gasparic, M. (2020). Bug density analysis in software reuse: Metrics and industrial case study. Empirical Software Engineering, 25(5), 3580–3605. https://doi.org/10.1007/s10664-020-09877-0

Hair, J. F., Hult, G. T. M., Ringle, C. M., & Sarstedt, M. (2021). A primer on partial least squares structural equation modeling (PLS-SEM) (3rd ed.). SAGE Publications.

Kumar, R., Rana, N. P., & Singh, M. (2022). Adoption of open-source software frameworks in agile environments: A resource-based perspective. Information Systems Frontiers, 24(1), 17–35. https://doi.org/10.1007/s10796-020-10059-0

Lopes, F., de Oliveira, D., & Amaral, L. A. (2020). Software reuse in the context of DevOps and microservices. Journal of Systems and Software, 161, 110467. https://doi.org/10.1016/j.jss.2020.110467

Mani, D., Mishra, D., & He, F. (2021). Strategic reuse of software components in large-scale systems: A value-based analysis. Decision Support Systems, 142, 113467. https://doi.org/10.1016/j.dss.2020.113467

Munir, H., Papatheocharous, E., & Petersen, K. (2023). Design pattern adoption and its impact on software quality: A systematic mapping. Information and Software Technology, 153, 107059. https://doi.org/10.1016/j.infsof.2022.107059

Park, J., Kim, H., & Choi, M. (2021). A study on software component reuse strategy based on machine learning prediction models. Applied Sciences, 11(2), 567. https://doi.org/10.3390/app11020567

Rani, N., Sharma, P., & Malhotra, R. (2021). Framework-based software reuse and software development effort estimation. International Journal of Information Technology, 13, 239–247. https://doi.org/10.1007/s41870-020-00527-7

Shah, M. A., Khan, A., & Khan, S. (2021). Framework-based development and its effects on software quality metrics: An empirical evaluation. Journal of Computer Languages, 63, 101049. https://doi.org/10.1016/j.cola.2021.101049

Sun, Y., Xu, Y., & Chen, Y. (2022). Framework-oriented reuse and agile release planning in cloud-native applications. Future Generation Computer Systems, 126, 107–118. https://doi.org/10.1016/j.future.2021.07.017

Wang, Z., Jiang, W., & Zhou, X. (2023). Improving software development lifecycle with reuse-oriented practices: A system dynamics approach. Software: Practice and Experience, 53(4), 817–834. https://doi.org/10.1002/spe.3143

Zhao, Y., & Yang, L. (2023). Understanding the productivity benefits of software reuse: Evidence from development logs. Empirical Software Engineering, 28(1), 12. https://doi.org/10.1007/s10664-022-10126-9.