A Comparative Analysis of Software Quality and Development Cost for a Community Elderly Database System A Case Study Comparing Four Development Approaches Using the Personal Software Process (PSP)

Wacharapong Nachiengmai[1], Danuphon Wunchaisatira[2], Satianpong Yodnin[3], Irin Somboon[4]

[1]Department of Software Engineering, Faculty of Engineering and Technology, North-Chiang Mai University, Chiang Mai, Thailand, wacharapong@northcm.ac.th

[2]Department of Software Engineering, Faculty of Engineering and Technology, North-Chiang Mai University, Chiang Mai, Thailand, danuphon@northcm.ac.th

[3]Department of Software Engineering, Faculty of Engineering and Technology, North-Chiang Mai University, Chiang Mai, Thailand, satianpong@northcm.ac.th

[4]Department of Software Engineering, Faculty of Engineering and Technology, North-Chiang Mai University, Chiang Mai, Thailand, aj.irin@gmail.com

Abstract

As Thailand fully transitions into an aging society, the development of information technology to support elderly care has become essential, particularly for safety-critical functions such as location tracking and emergency alerts. This research aims to analyze and compare software quality and development costs using the Personal Software Process (PSP) framework. The study compares four development approaches: (G1) Solo Waterfall, (G2) Pair Programming, (G3) Code Review, and (G4) Test-Driven Development (TDD). The experiment involved third-year software engineering students developing three core functions: health record management, Google Map location tracking, and emergency help requests, utilizing tools such as Express JS, MySQL, Mocha, and Chai.

Empirical results indicate that G4 (TDD) achieved the highest software quality, with a Defect Removal Efficiency (DRE) of 100% in critical functions and the lowest Defect Density (12.93-22.73 per KLOC), despite higher initial time costs due to test case development. Conversely, G1 (Waterfall) exhibited the lowest initial labor cost (1,585 THB) but recorded the highest post-run defects, leading to significant long-term hidden costs. G2 recorded the highest human resource costs as it required two developers for a single task. The suitability analysis concludes that TDD and Code Review are the most

appropriate approaches for safety-critical systems to ensure data integrity and the life safety of the elderly.

Keywords: Software Quality, Development Cost, Personal Software Process (PSP), Aging Society, TDD

Background and Statement of the problem

At present, Thai society has fully transitioned into an aged society. According to the report on the situation of the elderly in Thailand, the proportion of the population aged 60 and over is continuously increasing, marking Thailand's entry into a complete aged society. Consequently, the government and local administrative organizations must prepare service systems and data management to systematically support elderly care at the community level (Department of Older Persons, 2024). Specifically, the application of information technology is crucial to support efficient and timely monitoring, care, and assistance for the elderly.

Furthermore, such systems involve the collection and processing of health and personal data, requiring high levels of data accuracy and information security. Several software development process research papers indicate that software with numerous defects often leads to long-term hidden costs, including maintenance expenses, bug fixes, or risks to user trust—particularly in systems related to human safety.

However, software development for government agencies or local administrative organizations in the Thai context often faces budget and human resource constraints. As a result, many projects opt for traditional software development processes, such as the Waterfall model. Although it offers low initial costs and ease of planning, software engineering research suggests that this process may lead to high Costs of Quality (CoQ) in the later stages. This is because defects are typically detected toward the end of the project, causing the cost of remediation to increase significantly.

Over the past decade, the Personal Software Process (PSP), developed by Watts S. Humphrey, has been recognized in the software engineering field as a personal-level development framework that focuses on systematic planning, tracking, and quality control (Humphrey, 1995; 2005). PSP emphasizes managing the developer's workflow at

each stage—from estimation and design to coding, testing, and result review. Recent studies, such as those by Mansoor et al. (2017) and López-Martín et al. (2017), indicate that implementing PSP fosters development discipline, reduces errors caused by uncontrolled workflows, and supports the development of higher-quality software under time and resource constraints.

Additionally, recent trends in software process research reflect that PSP can be appropriately applied alongside various development methodologies and is not limited to traditional processes. Applying PSP in this manner strengthens the systematic approach of developers and supports early defect reduction, which significantly decreases the burden of late-stage troubleshooting. This concept aligns with software development for an aged society, which requires systems that are accurate, reliable, and capable of cost management within the constraints of government agencies and communities.

Accordingly, this research utilizes the Personal Software Process (PSP) as a framework for empirical data collection to compare four software development models: (1) Individual Waterfall development, (2) Pair Programming, (3) Team Code Review, and (4) Test-Driven Development (TDD). The evaluation is conducted through key performance indicators, namely Defect Removal Efficiency (DRE), Review Yield, and Cost of Quality (CoQ).

To enhance modernity and alignment with the nature of elderly care systems, this research also incorporates security defect auditing alongside functional defect tracking, as the system involves the storage of personal data and location coordinates. The development utilizes standard tools, including Express.js and MySQL, with Mocha and Chai for automated testing. The resulting analysis is expected to serve as academic data to support decision-making in selecting the most cost-effective software development process for community-level aged society software projects and to provide a reference guideline for government agencies in the future.

Objectives

1. To analyze and compare the software quality and development costs of a community-based elderly database system across different software development models, utilizing empirical metrics based on the Personal Software Process (PSP) framework.

2. To evaluate the efficiency of software defect prevention and removal, covering both functional defects and security vulnerabilities, by assessing Defect Removal Efficiency (DRE) and data integrity metrics.

3. To analyze the cost-effectiveness and suitability of various software development models in order to propose academic guidelines for selecting optimal software development processes for aged-society software projects within the context of government agencies and local communities.

Expected benefits

This research is expected to yield significant contributions in both academic and practical dimensions, as follows:

1. Academic Contributions: The study will generate a specialized body of knowledge regarding the correlations between software development processes, software quality, and development costs within the specific context of aged-society software. This will serve as a foundational academic reference for future software engineering research.

2. Practical Application and Implementation: The research findings will provide a definitive guideline for selecting appropriate software development methodologies for mission-critical systems operating under the budgetary and resource constraints typical of government agencies and local communities.

3. Security and Human Resource Development: The research promotes the development of highly accurate and reliable software while fostering quality control discipline and systematic development skills among software practitioners.

Conceptual Framework

1. Theoretical Framework and Research Concepts

This research integrates core Software Engineering principles to establish an academic framework for evaluating software quality and development costs, detailed as follows:

1.1 Personal Software Process (PSP)

The Personal Software Process (PSP) serves as the primary conceptual framework for systematic empirical data collection at the individual developer level. PSP

emphasizes the measurement, analysis, and quality improvement of software development by recording real-world data throughout the development lifecycle, including time, effort, and defects. This enables software engineers to control and enhance both quality and cost through a verifiable academic approach, as defined in the Personal Software Process Body of Knowledge (PSP BOK) (Pomeroy-Huff et al., 2009).

To evaluate process effectiveness, this study utilizes key metrics aligned with the PSP framework:

- Defect Removal Efficiency (DRE): Reflects the process's ability to detect and resolve errors before system delivery.
- Review Yield: Represents the proportion of defects discovered during the review phase prior to compilation or testing.
- Cost of Quality (COQ): Evaluated by the ratio of time spent on Prevention and Appraisal activities compared to the time spent on Failure (rework).

1.2 Software Development Approaches

The research employs four primary software development methodologies as independent variables:

- Waterfall (Solo Development)
- Pair Programming
- Team Code Review
- Test-Driven Development (TDD)

Each approach possesses distinct quality control characteristics. This study aims to compare the outcomes of each model under the same PSP measurement framework to provide a concrete analysis of differences in quality and cost.

1.3 System Criticality

The case study system is categorized as a Safety-Critical System, as it involves elderly data and services directly impacting user safety and well-being. This includes high-reliability functions such as location tracking and geocoding via map services. Furthermore, the Data Integrity of health records is a vital component of system criticality in this context.

1.4 Data Security

This research considers data security in conjunction with Functional Quality. It analyzes Security Defects occurring during development, such as access control management for sensitive personal and health data. Integrating security into the PSP framework allows for a comprehensive evaluation of system quality from both technical and ethical perspectives.

2. Research Framework

The research framework focuses on studying the impact of different software development models on quality and development costs.

2.1 Independent Variables: Four experimental groups:

- G1: Waterfall (Solo Development)
- G2: Pair Programming
- G3: Team Code Review
- G4: Test-Driven Development (TDD)

2.2 Dependent Variables (Quality): The number and types of Defects detected at each development stage and the functional accuracy of high-criticality features.

2.3 Dependent Variables (Cost Outcomes): Development Time and Effort (Man-hours) expended within each model.

Experimental Context: The system is developed by third-year Software Engineering students with similar programming and design backgrounds to control for the confounding variable of developer experience.

3. Research Hypotheses

The researcher has established the following comparative hypotheses:

Table 1: Comparative Hypotheses Across the Four Experimental Groups

Experimental Group	Software Quality Hypothesis	Cost/Effort Hypothesis
G1: Waterfall	Highest number of defects found Post-run.	Lowest cost in terms of Man-hours.
G2: Pair Programming	High defect reduction capability from the start of the coding process.	High cost due to utilizing two human resources for a single task.
G3: Code Review	Highly effective at detecting Pre-run errors.	Moderate cost, with increased time allocated to the review phase.
G4: TDD	Highest code quality with minimal bugs.	High initial cost due to the Learning Curve of writing test suites.

Based on the comparative hypotheses of the four experimental groups, it is evident that the differing software development approaches significantly impact both software quality and development cost/time. Group G1 (Waterfall) tends to encounter a high volume of post-run defects, despite having the lowest cost in terms of man-hours. Conversely, Group G2 (Pair Programming) and Group G3 (Code Review) demonstrate the capability to mitigate errors during the pre-run phase, albeit at the expense of varying levels of increased costs.

Group G4 (Test-Driven Development: TDD) is projected to yield the highest software quality with the fewest defects, as testing is integrated from the very beginning of the development process. However, the TDD approach incurs higher initial costs and time requirements due to the learning curve associated with the methodology. Collectively, these hypotheses clearly reflect a trade-off relationship between software quality and the development cost/effort required by each respective development model.

Research Methodology

This study is an experimental research designed to compare the software quality and development costs of an elderly database system under the framework of the Personal Software Process (PSP). The operational details are as follows:

1. Participants and Background

The sample group consists of 10 third-year Software Engineering students who have successfully completed relevant foundational courses, including Computer Programming, Object-Oriented Programming, and Object-Oriented Analysis and Design. The participants were purposively selected and divided into four experimental groups as follows: G1 (n=2), G2 (2 pairs, n=4), G3 (n=2), and G4 (n=2). This ensures the participants possess the necessary readiness to develop software characterized by high complexity and criticality, specifically regarding data accuracy, system reliability, and security.

2. Experimental Groups

The researcher divided the 10 participants into four experimental groups to compare the outcomes of the different development processes. Each group performed their assigned tasks under the systematic data collection framework of the PSP. The distribution of participants is as follows:

Group G1: Individual Waterfall development (Solo), consisting of 2 participants working independently as a baseline for cost and defects.

Group G2: Pair Programming, consisting of 2 pairs (4 participants total) to study the impact of collaborative coding on defect reduction.

Group G3: Development utilizing the Code Review process, consisting of 2 participants to measure the efficiency of peer verification and Review Yield.

Group G4: Test-Driven Development (TDD), consisting of 2 participants focusing on high-quality code through pre-coding test case development.

3. Measurement and Scope

The evaluation utilizes identical criteria across all groups. Software quality is assessed based on the number of Defects, Defect Removal Efficiency (DRE), and Security Defect audits. Development costs are measured through Development Time and the Cost of Quality (COQ) to reflect the overall cost-effectiveness of the software development process.

4. Development and Testing Tools

System development utilizes Express.js as the primary backend framework and MySQL as the database management system for storing personal data, health records, and location coordinates. Software testing is conducted using Mocha and Chai to support automated testing, software quality data collection, and the measurement of key metrics according to the PSP framework.

5. System Functions for Data Collection

The researcher mandated that all four experimental groups develop software covering three core functions, each with varying levels of importance and criticality:

(1) Elderly & Health Data Management: This encompasses the recording, editing, and retrieval of personal and health data. The focus is on evaluating data accuracy, database schema consistency, and security defect auditing regarding data access through the MySQL management system.

(2) Google Maps Location Tracking: This function requires high precision and reliability. The researcher utilizes this function to collect data on the number of defects arising from calculation logic, coordinate management, and integration with external services, reflecting software quality in scenarios where data discrepancy poses a risk.

(3) Emergency Help Requests: This is the most critical function, as it directly impacts user safety and life. This function is employed to evaluate Defect Removal Efficiency by utilizing DRE and Review Yield metrics to compare the capability of each development approach in identifying and eliminating errors at an early stage, prior to actual system deployment.

Research Results

Based on the empirical data analysis following the Personal Software Process (PSP), the research findings and discoveries corresponding to the objectives and hypotheses are reported as follows:

Table 2: Empirical Results Based on the Personal Software Process (PSP)

Function	Group	LOC	Time (Min)	PD	Defects Found		Defect Density	DRE (%)
					Pre-Compile	Post-Compile		
1. Elderly Data Management	G1	134.50	126.89	1.06	2.01	11.04	97.01	15.38
	G2	138.25	98.05	1.41	6.01	7.01	94.20	46.15
	G3	108.50	144.67	0.75	7.03	2.01	83.33	77.78
	G4	232.50	169.71	1.37	3.01	0.00	12.93	100.00
2. Google Maps Tracking	G1	82.50	93.75	0.88	2.01	17.10	231.71	10.53
	G2	94.25	114.94	0.82	5.01	5.01	106.38	50.00
	G3	86.50	133.08	0.65	5.03	3.02	93.02	62.50
	G4	132.50	179.05	0.74	2.01	1.00	22.73	66.67
3. Emergency Help Requests	G1	55.50	99.11	0.56	2.02	7.06	163.64	22.22
	G2	54.25	87.50	0.62	3.01	2.01	92.59	60.00
	G3	72.50	127.19	0.57	3.02	1.01	55.56	75.00
	G4	111.50	168.94	0.66	2.01	0.00	18.02	100.00

Note: The data presented in Table 2 (LOC, Time, and Defects) represent the Average (Mean) values calculated from each experimental group.

Description of Key Research Metrics

The researcher utilized primary metrics from the Personal Software Process (PSP) as performance indicators. The definitions and formulas are as follows:

1. Productivity (PD): The rate of software production relative to time, measured in LOC/Min. It compares the cost-effectiveness of time resources across experimental groups.

$$Productivity = \frac{Lines\ of\ Code(LOC)}{\text{Development Time (Minutes)}}$$

2. Defect Density (DD): The ratio of defects found per unit of software created, serving as an indicator of code "cleanliness" and quality. A lower DD signifies reduced operational risk, which is critical for high-criticality functions.

$$\text{Defect Density} = \left(\frac{\text{Total Defects}}{\text{Lines of Code}}\right) \times 1{,}000$$

Note: The multiplier of 1,000 is a standard software engineering convention for easier comparison.

3. Defect Removal Efficiency (DRE%): The ability of the development process to detect and resolve errors before system execution (pre-compile). It measures how effectively an approach prevents errors from reaching the end-user, thereby reducing Hidden Costs.

$$\text{DRE\%} = \left(\frac{\text{Pre} - \text{Compile Defects}}{(\text{Pre Compile Defects} + \text{Post Compile Defects})}\right) * 100$$

Analysis of Findings

1. Comparison of Quality and Cost

Group G1 (Waterfall) exhibited the lowest DRE% across all functions (reaching a minimum of 10.53% in the Google Maps function) and the highest Defect Density (DD) of 231.71 in the tracking function. This aligns with the hypothesis that this method encounters the most errors post-run. Conversely, Group G4 (TDD) demonstrated significantly superior quality, achieving a DRE% of 100% in data management and emergency requests. Although its Productivity (PD) was not markedly higher than other groups, it

resulted in the lowest Defect Density (12.93-22.73), validating the hypothesis regarding peak quality.

2. Efficiency in Critical Functions and Data Integrity

In the Google Maps Location Tracking function, which requires High Reliability, Group G4 (TDD) and Group G3 (Code Review) successfully maintained a DD below 100, while G1 and G2 showed significantly higher error density. Regarding Data Integrity and security, G4 detected all defects via Mocha and Chai before delivery, resulting in zero leakage of defects in health record management.

3. Cost-Effectiveness and Suitability Analysis

Although Group G2 (Pair Programming) achieved the highest PD in the first function (1.41 LOC/Min), the total cost-effectiveness is lower when considering the two-person requirement compared to G3 or G4, which achieved higher DRE% with a single developer. For aged-society software where safety is paramount, TDD (G4) offers the highest Value for Money in the long term by mitigating life-threatening defect risks.

4. Behavioral and Learning Outcomes

Empirical evidence suggests that participants utilizing PSP alongside modern tools significantly reduced final-stage errors when adopting Code Review or TDD approaches.

In addition to quality metrics, Development Cost Analysis and cost-effectiveness are vital components that complete the overall picture of this research. The researcher established a labor rate for software development at 300 THB per hour (or 5 THB per minute). Actual time spent developing the three core functions was used to calculate the total development cost for each experimental group based on their respective work models. These calculations facilitate a comparative analysis of the relationship between development processes, software quality, and cost-effectiveness, as presented in the following table:

Table 3: System Development Costs

Group	Development Model	Total Time (3 Functions)	Number of Developers	Total Labor Cost (THB)
G1	Waterfall (Solo)	317 Min.	1	1,585
G2	Pair Programming	300 Min.	2	3,000
G3	Code Review	402 Min.	1	2,010
G4	TDD	516 Min.	1	2,580

Analysis of Development Costs

Table 3 provides a comparison of the development costs for the elderly database system across the four experimental groups, highlighted by the following points:

1. Lowest Initial Cost (G1: Waterfall)

Group G1 incurred the lowest labor cost at 1,585 THB, validating the hypothesis that individual Waterfall development is cost-efficient regarding man-hours. However, considering the high volume of post-compile defects (35 total: 11+17+7), this group faces the highest risk of significant Hidden Costs for maintenance and rework in the future.

2. Highest Human Resource Cost (G2: Pair Programming)

While Group G2 recorded the lowest development time per function (300 minutes total), the requirement for two developers working simultaneously resulted in a doubled labor cost of 3,000 THB. This represents the highest cost in this experiment, consistent with the initial hypothesis.

3. Investing in Quality and Value (G4: TDD and G3: Code Review)

Group G4 (TDD) incurred a cost of 2,580 THB, higher than G1 and G3 due to the learning curve and the additional time required to develop test suites. However, in terms of Value for Money, this method provides the highest quality with only one defect escaping to the post-run phase, thereby drastically reducing long-term failure costs. G3 (Code Review) offers a strategic balance; with only a marginal cost increase over G1 (2,010 THB), it significantly improves Defect Removal Efficiency (DRE).

4. Cost of Quality (COQ)

Analysis Aligned with the PSP framework, this research demonstrates that increasing the time allocated to Prevention and Appraisal activities (as seen in G3 and G4) effectively manages high-criticality systems, such as location tracking and health records, despite the higher initial investment.

Summary of the Study

Based on the comparative analysis of the four development models (G1–G4) across three system functions with varying levels of criticality, the key findings of this study are concluded as follows:

1. Software Quality and Reliability Group G4 (TDD) yielded the highest quality results, demonstrating the lowest Defect Density across all functions (ranging from 12.93 to 22.73 per KLOC). Furthermore, it achieved a DRE of 100% in the Data Management and Emergency Help Request functions, indicating that no defects leaked into the actual execution phase. In contrast, Group G1 (Waterfall) exhibited the lowest quality, recording the highest number of post-compile defects in every function, particularly in the Google Maps tracking function, where the DRE was as low as 10.53%.

2. Development Cost and Effort As hypothesized, Group G1 incurred the lowest initial labor costs; however, it carries a significant risk of high hidden costs due to the extensive rework required to fix numerous post-development defects. Group G2 (Pair Programming) presented the highest human resource costs, as it utilizes two developers for a single task, even though it reduces defects more effectively than the traditional method. While Group G4 (TDD) required a higher initial time investment for writing test suites, it offers the highest long-term cost-effectiveness for safety-critical systems.

3. Suitability Analysis For functions directly impacting human safety—such as Location Tracking and Emergency Help Requests—the TDD (G4) and Code Review (G3) models are the most suitable. These approaches provide the high reliability and data integrity necessary for mission-critical community applications, ensuring that the system remains accurate and trustworthy under high-risk scenarios.

Discussions

The analysis of the empirical data collected through the Personal Software Process (PSP) for the development of the elderly database system yields the following points of discussion:

1. Software Quality and Defect Removal Efficiency

The findings clearly demonstrate that Group G4, utilizing Test-Driven Development (TDD), produced the highest quality software, achieving a Defect Removal

Efficiency (DRE) of 100% in key functions, specifically Elderly Data Management and Emergency Help Requests. This outcome aligns with the theories of Humphrey (1995, 2005), which posit that PSP is a process centered on early defect prevention through systematic planning, inspection, and personal-level measurement. Furthermore, it supports the PSP Body of Knowledge (Pomeroy-Huff et al., 2009), which highlights that structured quality assurance prior to coding and phased testing significantly enhances defect removal capabilities.

2. Balancing Cost and Quality

When evaluating the dimensions of cost alongside software quality, Group G1—despite having the lowest initial labor costs—encountered a high volume of late-stage defects. This led to significant hidden costs in maintenance, rework, and a reduction in overall system reliability. This observation is consistent with Humphrey (2005) and Pomeroy-Huff et al. (2009), who suggest that the true cost of software should be measured through the Cost of Quality (COQ) rather than initial development expenses alone. Regarding Group G2 (Pair Programming), while it reduced defects more effectively than G1, it required doubling human resource costs. This mirrors the findings of Mansoor et al. (2017), which suggest that while increased human resources can reduce errors to an extent, they cannot fully replace structured inspection processes like PSP or TDD. These findings reinforce the concept that early investment in quality assurance yields superior long-term returns.

3. Suitability in the Context of Safety-Critical Systems

The results for functions directly related to safety, such as Location Tracking and Emergency Help Requests, confirm that TDD (G4) and Code Review (G3) are the most appropriate methodologies for such systems. Conducting Security Defect audits during the development phase further strengthens confidence in protecting sensitive elderly health data. This is in accordance with the Department of Older Persons (2024), which emphasizes the necessity for secure and reliable information systems to support Thailand's rapidly expanding aged society. These security considerations are inherently linked back to the long-term quality and cost-effectiveness of the system.

4. Impact on Personnel Development and Professional Standards

The application of the PSP framework to third-year students with foundations in programming and Object-Oriented Design shows that training within this framework fosters systematic software quality control skills. This is consistent with studies by Mansoor et al. (2017) and López-Martín et al. (2017), which found that integrating PSP into academic curricula elevates students' abilities in planning, inspection, and performance evaluation through concrete metrics. Moreover, the PSP Body of Knowledge (2009) identifies developer discipline in recording and analyzing personal work data as a cornerstone for raising professional standards. The findings suggest that integrating PSP into software engineering education can serve as a blueprint for establishing development standards for local government agencies, particularly for systems affecting the health and quality of life of the elderly, aligning with Thailand's current social and policy context.

Due to the small sample size per group (n=1 or 2 units per group), this study is identified as a preliminary empirical case study. While the findings provide significant insights into the PSP process, they represent initial empirical evidence within a controlled academic environment.

Recommendations

1. Policy and Practical Implications

The findings of this research can serve as a strategic guideline for establishing policies and procurement requirements for software development within government agencies and local administrative organizations. For high-criticality systems involving sensitive personal data, it is recommended that development mandates include quality-centric processes, such as Test-Driven Development (TDD) or Code Review, to mitigate security risks and system failures. Furthermore, budgetary planning should shift toward evaluating the Total Cost of Ownership (TCO) rather than focusing solely on initial development costs. This holistic approach will help organizations avoid significant long-term hidden costs associated with maintenance and error remediation.

2. Directions for Future Research

Future studies should extend the scope of analysis to investigate long-term hidden costs, specifically focusing on maintenance expenditures and the socio-economic risks arising from system failures. Additionally, future research could incorporate deeper security assessments, such as Penetration Testing (Pentest), to ensure robust data protection. Expanding the research context to other platforms, such as mobile applications specifically designed for the elderly, would also enhance the comprehensiveness and practical relevance of the findings in real-world scenarios.

References

Department of Older Persons, Ministry of Social Development and Human Security. (2024). *Situation of the Thai older persons 2023*.

Humphrey, W. S. (1995). *A discipline for software engineering*. Addison-Wesley.

Humphrey, W. S. (2005). *PSP: A self-improvement process for software engineers*. Addison-Wesley.

López-Martín, C., Nassif, A. B., & Abran, A. (2017). A training process for improving the quality of software projects developed by a practitioner. *Journal of Systems and Software*, *131*, 98–111. https://doi.org/10.1016/j.jss.2017.05.050

Mansoor, S., Bhutto, A., Bhatti, N., Patoli, N. A., & Ahmed, M. (2017). Improvement of students' abilities for quality of software through personal software process. In *2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT)*. IEEE. https://doi.org/10.1109/ICIEECT.2017.7916550

Pomeroy-Huff, M., Cannon, R., Chick, T. A., Mullaney, J. L., & Nichols, W. (2009). *The Personal Software Process (PSP) body of knowledge, version 2.0* (CMU/SEI-2009-SR-018). Software Engineering Institute, Carnegie Mellon University.