INFOTECH
Journal of Applied Sciences, Informatics, Computing and Knowledge Society

Volume 4, No. 2, July/December, 2023

Adjustment of Models and Procedures for Web Development

Noah Quinn[1]

[1] Universidad de Middlesex - Inglaterra, quinnnoah460@gmail.com, https://orcid.org/0009-0002-1829-9496

ABSTRACT

In the realm of Software Development, technology poses a dynamic challenge, constantly reshaping the Software Engineering landscape. This article explores the authors' efforts in adapting various methodologies and processes in Software Engineering, specifically tailored for Web Development. While the Personal Software Process, originally formulated by [5], was designed for third-generation languages, it has undergone adjustments to align with the contemporary technological landscape. Many companies continue to seek certification in this process for software development. The article details the adaptation of this process in twenty-five projects, incorporating Modular Programming, Model View Controller, and Agile Methodologies. The presented results not only highlight the adaptation for Web Development but also emphasize the incorporation and modification of activities aimed at ensuring the overall quality of the development process and the final product.

Keywords: *Technology; Software Engineering; Web Development; Adaptation*

Received:
12 July, 2023

Accepted for publication:
25 August, 2023

Ajuste de Modelos y Procedimientos para el Desarrollo Web

RESUMEN

En el ámbito del Desarrollo de Software, la tecnología plantea un desafío dinámico, dando forma constantemente al panorama de la Ingeniería de Software. Este artículo explora los esfuerzos de los autores en adaptar diversas metodologías y procesos en Ingeniería de Software, específicamente diseñados para el Desarrollo Web. Mientras que el Proceso de Software Personal, originalmente formulado por [5], fue diseñado para lenguajes de tercera generación, ha experimentado ajustes para alinearse con el panorama tecnológico contemporáneo. Muchas empresas continúan buscando la certificación en este proceso para el desarrollo de software. El artículo detalla la adaptación de este proceso en veinticinco proyectos, incorporando Programación Modular, Modelo Vista Controlador y Metodologías Ágiles. Los resultados presentados no solo resaltan la adaptación para el Desarrollo Web, sino que también enfatizan la incorporación y modificación de actividades destinadas a garantizar la calidad general del proceso de desarrollo y del producto final.

Palabras clave: *Tecnología; Ingeniería de Software; Desarrollo Web; Adaptación*

INTRODUCTION

The conventional Waterfall Model [11] has undergone subtle modifications leading to the emergence of specific methodologies tailored for distinct application domains upon integration with contemporary Software Development and Engineering technologies. [5], referencing the Personal Software Development Process (PSP), asserts that the PSP is a personalized approach adaptable to meet the requirements of individual developers. This process is not confined to any particular programming or design methodology, making it compatible with diverse approaches, including Agile software development. Software Engineering methodologies exhibit a spectrum from predictive to adaptive, with PSP categorized as predictive and Agile as adaptive. Despite their dissimilarities, TSP/PSP and Agile methodologies share common concepts and approaches, particularly in terms of team organization.

Thus, it becomes apparent that the framework established in PSP can seamlessly integrate with various methodologies, whether traditional or agile. Furthermore, these methodologies can coexist by adjusting their processes to accommodate development requirements. This study introduces a novel empirical and experimentally validated proposition aimed at enhancing and innovating within the PSP methodology. The findings are then applied to the development process within the Web environment through an adaptation to the MVC Layered Model (Model View Controller) [2], employing modular programming as an agile methodology.

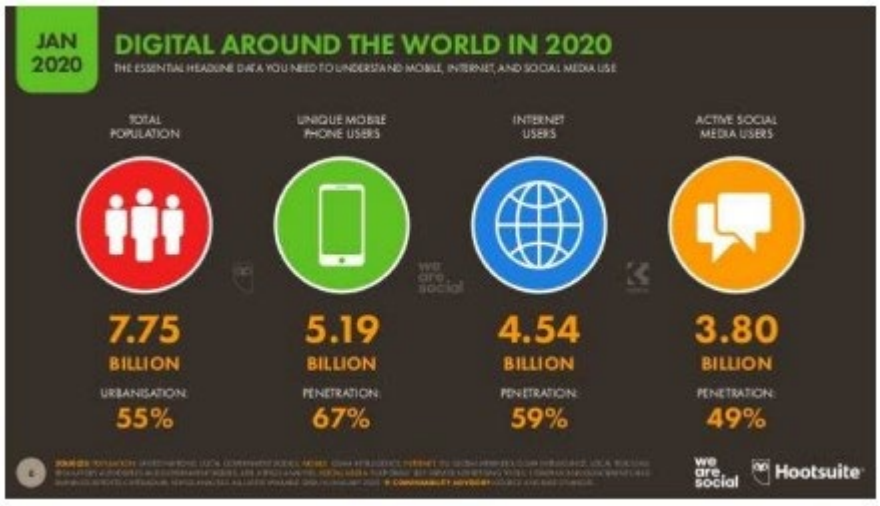


Figure 1 illustrates the utilization of Digital Development

The aim of this study is to introduce enhancements to the PSP activities by integrating the MVC architecture and employing modular programming as an agile methodology across twenty-four web development projects. The evaluation encompasses the analysis of performance, productivity, and the Cost of Quality.

The significance of this project lies in the innovative application of transforming a conventional line-of-code-based PSP process into a functional, modular approach under the MVC framework, tailored to a crucial sector such as web development, as depicted in Figure 1.

Global internet user penetration levels, as indicated by Wearesocial studies in January 2020, stand at 59%, marking a 14% increase from the previous year. Within web software development, responsive web design is employed to ensure access for mobile users, constituting 67% of the population with a 28% annual growth, as illustrated in the preceding graph.

Software quality has been a topic of debate, but since 1986, [5] initiated a project that revolutionized Software Engineering. The outcome is a process enabling the measurement of Software Development Quality, thus determining the quality of the final product. Adopting the fundamental principle that asserts, "If there was quality during the software development process, the finished product will bear that quality seal" [5].

The proposed methodology offers the advantage of incremental and iterative quality improvement. With each execution and refinement, the process and its deliverables experience enhancement in times, sizes, and libraries. This leads to a cascade of benefits, culminating in the accumulation of historical data for future estimates.

Problem Statement

The challenge at hand revolves around the absence of clear and specific methodologies tailored to emerging technologies. Currently, various frameworks have emerged by adapting standards designed for desktop software development for use on the web. However, this approach leaves significant knowledge gaps that are well-addressed in desktop software but overlooked in web software. Consequently, there is a lack of an appropriate software methodology to effectively address the needs of these emerging technologies, exemplified by [10] as a project control solution.

In their quest for improved project organization, many software development companies resort to combining different methodologies to create novel working approaches. Nevertheless, each organization adapts these methodologies in distinct ways, contributing to a fragmented landscape.

State of the Art

This paper takes a stance on the utilization of Personal Software Process (PSP) metrics [5]. It articulates how and why PSP metrics can be instrumental in teaching and learning software engineering. While acknowledging that PSP cannot solve every challenge faced by students and professionals in software development, it does serve as a guiding framework. The data and metrics offered by PSP lay the foundation for an analytical and scientific approach, showing promising potential for success compared to other methods or tools. Furthermore, these metrics endorse a more effective education paradigm, replacing traditional programming teaching with coaching that provides detailed, specific guidance on improvement [3].

The PSP, designed to enhance the quality, predictability, and productivity of software engineers, addresses the improvement needs of individual engineers and small software organizations. Having been taught at various universities and introduced by industrial software organizations, PSP offers a defined sequence of process improvement steps coupled with performance feedback at each stage. This approach aids engineers in understanding the quality of their work and appreciating the effectiveness of their methods. Early experiences with PSP demonstrate a tenfold improvement in average test defect rates and a productivity increase of 25% or more [6].

Earlier studies on web design methodologies encompass the Relationship Management Methodology (RMM), Object-Oriented Hypermedia Design Method (OOHDM), and UML-Based Web, a model grounded in object-oriented techniques.

RMM, initially introduced in [8], has undergone various evolutions to meet the escalating demand for hypermedia applications on the World Wide Web. The updated methodology finds application in the design of rich web applications, delving into design and implementation aspects, including database integration. It explores both top-down and bottom-up approaches to the development of Web Information Systems (WIS). The revised RMM introduces new constructs, featuring graphical and

programming language notations. Overall, RMM advocates for robust design practices and the sustainable evolution of hypermedia.

OOHDM, the object-oriented hypermedia design method, presents a model-based strategy for constructing expansive hypermedia applications. This methodology has found application in the design of diverse applications, including websites, information systems, interactive kiosks, and multimedia presentations. It encompasses four distinct activities: Conceptual Design, Navigation Design, Abstract Interface Design, and Implementation. These activities are executed through a combination of incremental, iterative, and prototype-based development styles. The segregation of interface, navigation, and conceptual design as individual activities allows for a focused approach to different concerns, one at a time. This approach yields more modular and reusable designs, providing a framework for reasoning in the design process that encapsulates expertise specific to each activity. Additionally, the design primitives for interface design can be easily mapped to non-object-oriented implementation environments or languages (such as HTML or Toolbook). Consequently, OOHDM can be employed irrespective of whether the target system is purely object-oriented, a conventional environment, or a hybrid system commonly encountered on the Internet [12].

METHODS

The methodology involves the estimation of historical data through linear regression and the calculation of average standard deviations. Initially, these calculations were conducted for over twenty-five thousand lines of code, with more than sixty-two inserted programs serving as tests. Additionally, fifteen internal versions were corrected before the official release in [5].

Initiating methodological shifts from the conventional Web Development process, we transitioned towards the MVC methodology. This transformation was achieved by aligning with the PSP process and incorporating modular development. The outcomes highlight innovative modifications to the PSP process when applied to web pages, addressing challenges encountered during the adaptation of the MVC methodology and modular development.

For the proposed approach, the development comprised twenty-four modules within the web domain, with a cumulative effort of over one hundred and seventy delta hours invested in the PSP process version 2.1.

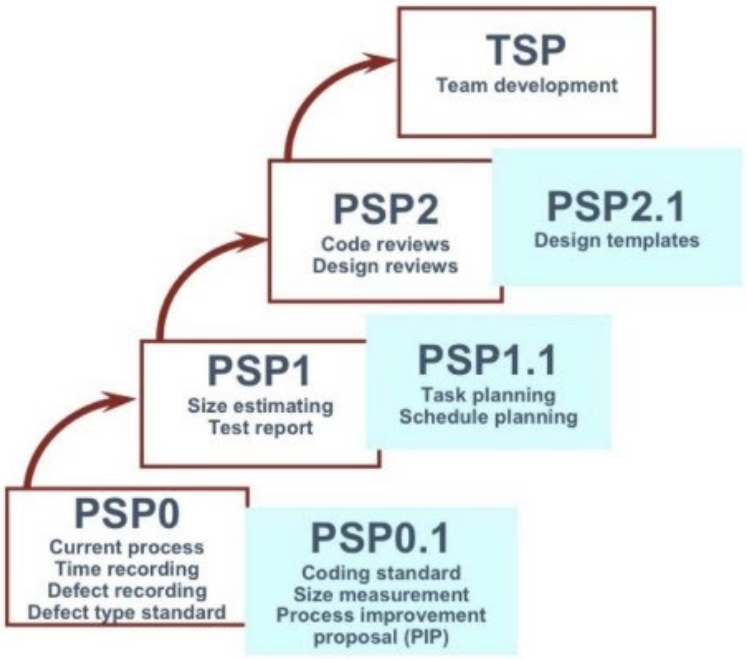


Figure 2 illustrates the conventional stages of software development

Historical Facts

The examination of the implementation is detailed in Section 1, compiled from historical data developed within the methodologies and processes presented in the initial section of the proposal.

Testing Methods for Size

Considering the developmental history outlined in Table 1, when generating a new module in the project, the estimations for this module are presented alongside the overall project of the system, elucidating the estimation process. The calculation for the estimation of code lines to be programmed for a 183 LOC module (Lines of Code) is 206 LOC in estimation method B, based on planning data, and 237 LOC in estimation method A, based on historical data. These exhibit a deviation of 88.2 and 90.4, respectively, as indicated in Figure 2. This implies that method B is the preferred option due to the precision of historical data entered in the planning stage, ensuring a more accurate estimate for the development of the next module.

Estimated Productivity

The productivity estimate for planned development sizes and times is 42.2 LOC/Hr, aligning with the current date's productivity of 35.2 LOC/Hr (± 19.5).

The Value Proposition of this Paper

The proposal is centered on modifying the traditional PSP process, initially designed for third-generation languages, by incorporating essential tasks and activities that align with and adapt to web development. A pivotal aspect is the accurate interpretation of historical data enabling estimations regarding the quality scope of the software. Various methodologies, including MVC, modular programming, and agile methodologies, will be employed to bring the user or client closer to the development process. The anticipated result is a novel and functional Software Development process tailored for the web domain, embodying both innovation and functionality.

METHODOLOGY

To initiate the adaptation to the development process, commence by sequentially applying the PSP stages 0, 0.1, 1, 1.2, 2, and 2.1, as illustrated in Figure 2. Throughout the process adaptation, modifications directly impacting web application development have been incorporated.

In the PSP framework, all tasks and activities essential for a software engineer during the software product development process are precisely delineated in a set of documents known as scripts. Adhering to these scripts in a disciplined manner is imperative, as the success of the sought-after improvement's hinges upon it (The Personal Software Process (PSP)).

The initial step in implementing the proposal involves applying the PSP 0 process. This entails daily software development while only augmenting the recording of development times in adherence to the Waterfall methodology. The collected data serves as a foundation for future estimates. During this phase, introducing changes in the design stage is optimal, such as incorporating Web design (Web View Design). This involves generating HTML tagging with responsive styles using Bootstrap and CSS3, or in accordance with the Design methodology.

In the MVC framework, this alteration involves creating the Web interface in the project's View folder without functionality, presenting it to the user for approval (User Review). Before progressing to coding,

user-requested changes are implemented in the design stage. This singular change serves to instill confidence and focus on time control.

Post-Compile stage, the generated module is presented to the client again for approval as part of the testing process. The subsequent step involves applying the PSP 0.1 process, enabling the addition of code line count in the development process. The objectives of PSP 0.1 encompass measuring program size, performing size counting, and ensuring precise measurements.

To approach the number of functions and obtain a more accurate count, it is advisable at this stage to add the draft of the module to be implemented (Sketchboard Web) in the requirements and obtain the ER diagram with functionalities during the requirements survey stage (Using PSP 0.1).

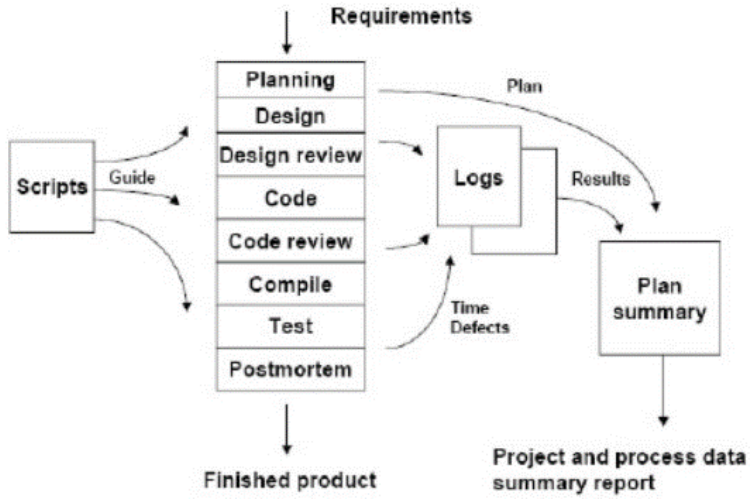


Figure 3 illustrates the software development phases in the Personal Software Process (PSP)

Similarly, it is recommended to incorporate the unit tests of the Programmer (Coder's Unit Test) during the coding phase. This involves verifying the programmed functionalities locally on localhost.

In PSP 1, the personal planning process is integrated, involving the establishment of a systematic and repeatable procedure for developing software size estimates (Using PSP 1, 2006). At this juncture, it is advisable to include the technical conceptual design of the software. This aids in illustrating the relationships among files to be used in development and assists in organizing the MVC and its functions. Optionally, this can also be adapted to the utilization of a Development Framework.

The subsequent phase in the Growth stage involves the application of PSP 1.1, which strives to introduce and implement methods for creating resource plans and schedules, tracking performance against these

plans, and determining probable project completion dates. Additionally, it is suggested to include the synchronization of the Database and the source files to the server during the compilation stage, in a designated section for tests. This enables validation by the user in the Test stage.

In the PSP 2 stage, two additional stages are introduced within the PSP numbering system, aiming to enhance development quality by identifying injected defects in the design and coding stages. It is recommended at this point to scrutinize the design and coding standards established in the PSP 0.1 process [13].

Moving to the PSP 2.1 stage, Design combined with UML is incorporated into the Development process. This phase introduces supplementary measures to manage process quality, along with design templates providing an organized framework and format for recording designs.

The subsequent enumeration outlines the components incorporated in this stage:

1. PSP 2.1 design review script,
2. PSP 2.1 design review checklist,
3. Operational specification template,
4. Functional specification template,
5. State specification template,
6. Logic specification template.

As part of the enhancements introduced in this stage for Web design, the deliverables encompass the Use Case Diagram and Specification, the sequence diagram, the State diagram, the Relational Diagram, the Data Dictionary, and the Database Design Data.

In summary, the activities and deliverables, including forms, templates, or standards stipulated by PSP in its various versions, as delineated in this section, are presented in Figure 4. Specifically, in version 2.1, verification of all deliverables from other versions is feasible.

Table 4 illustrates the designated preparation points for the deliverables based on the chosen PSP version. Furthermore, in Table 5, a correlation is observed with the previous table, categorizing the deliverables added to ensure functionality in Web development and uphold its quality, positioned on the right side of the table.

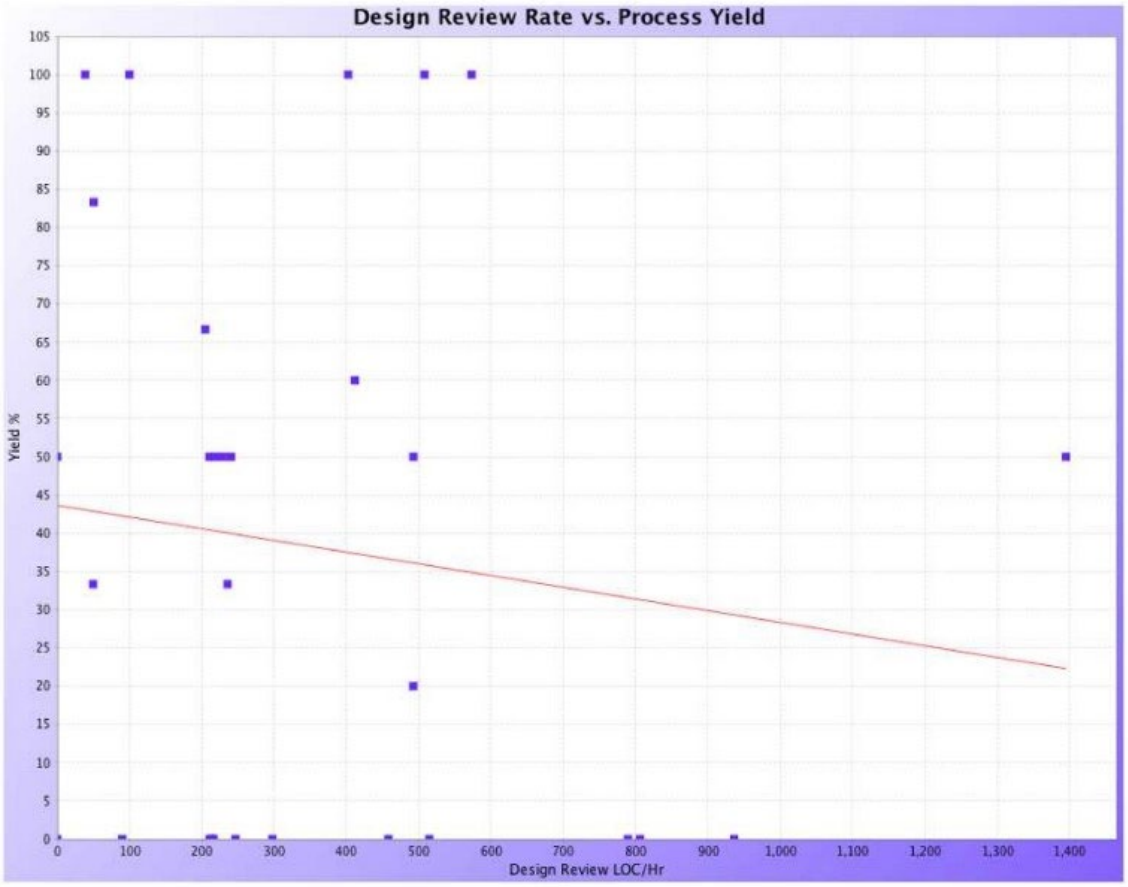


Figure 4 depicts the experimented process

Experimental Findings

The primary goals of PSP encompass maximizing software quality, instilling a culture of continuous improvement in the development process, enhancing the overall quality of development, and increasing productivity. Recognizing that quality relies on measurement, PSP incorporates a set of forms utilized for collecting essential metrics [1]. PSP establishes three fundamental metrics—size, time, and defects—and other measurements are derived from these foundational metrics. Defects, a metric linked to software quality, can be mitigated through the use of PSP, as noted in [1].

Results On Performance

Within the PSP, performance is assessed through two metrics: the percentage of total defects discovered and rectified within a stage, and Process Yield, denoting the percentage of defects addressed prior to the initial test, compilation, and testing phases [3].

As illustrated in Figure 4, performance statistics during the design review stage exhibit a negative slope, declining from 45% to 22%. This trend signifies a reduction in the number of defects as projects advance through the Design review stage, with a greater proportion of defects being addressed before reaching the testing phase.

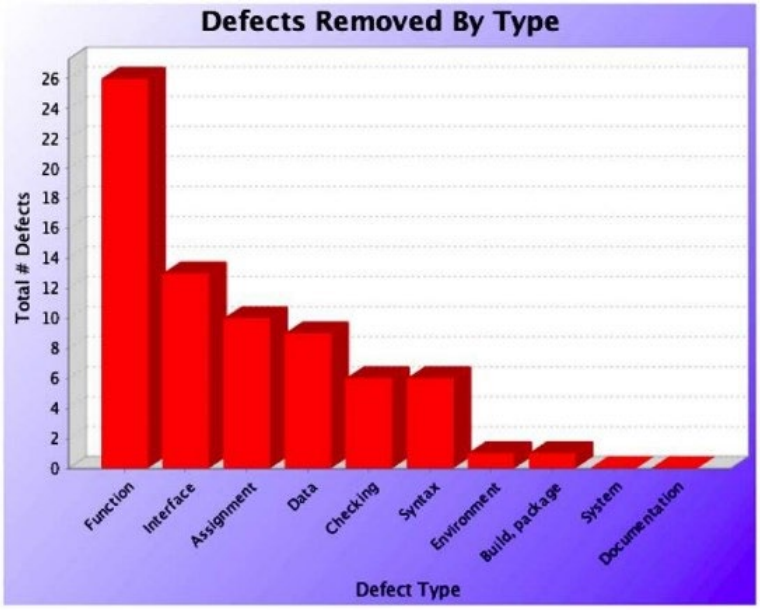


Figure 5 displays the inventory of failures

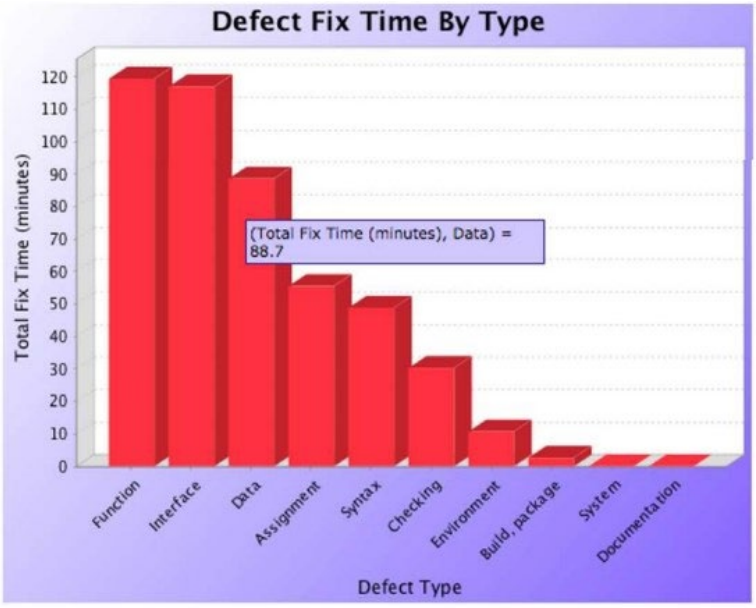


Figure 6 illustrates the time delay for resolving failures

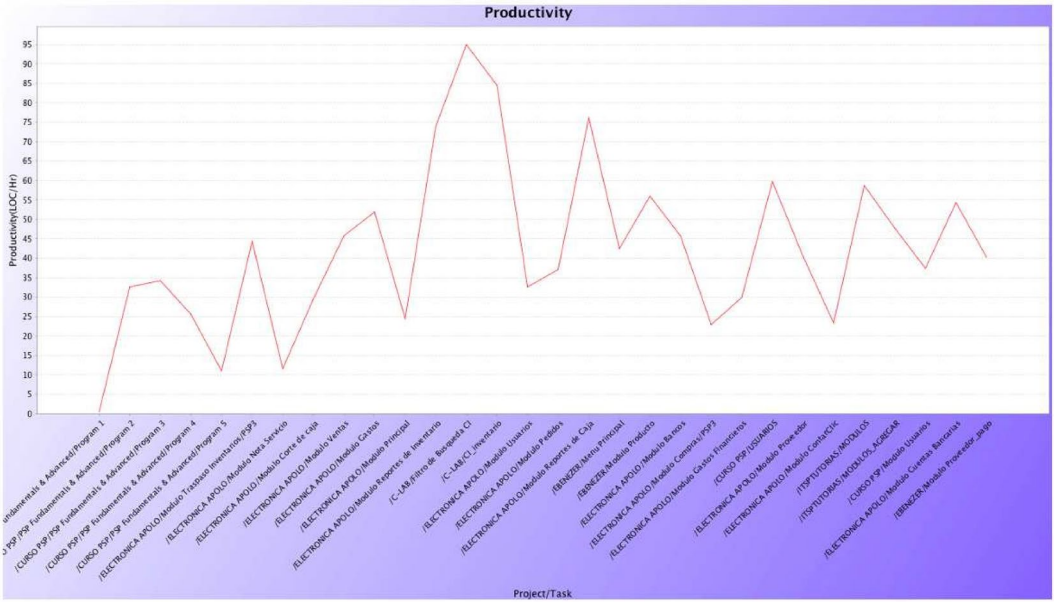
RESULTS

Results on Productivity

Productivity in PSP is quantified by the number of Lines of Code (LOC) developed per hour. As depicted in Figure 6, the average productivity exhibited a positive trend, escalating from 30 LOC/Hr. in the initial program to 40 LOC/Hr. Additionally, a distinction is noticeable between less complex modules, elevating productivity to over 50 LOC/Hr., and more intricate ones maintaining productivity up to 40 LOC/Hr. An exception is observed in the CI search filter module, registering a productivity rate close to 90 LOC/Hr. This anomaly is attributed to alterations in user requirements during the design stage. Productivity serves as a valuable self-assessment tool upon project completion, facilitating the determination of process improvement.

Results on Quality Cost

The relationship between the failure rate and quality cost parameters gauges the quality of the engineering process, aligning with Juran's quality improvement program principles [4]. This evaluation of quality cost reflects the percentage of development time allocated to quality assessment activities. Notably, from project 6 onward, an uptick in the cost of quality is observed due to an increase in defects during the testing or final development stages. However, in the last 10 projects, a marginal decline in the cost of quality is noted, primarily attributable to reduced defects in the final stages. This reduction is largely associated with the inclusion of the user in the Review Design stage.



DISCUSSION

Based on the obtained results, a discernible enhancement in the quality of development, the overall process, and its outcomes is evident through the amalgamation of PSP, Modular Programming, MVC, and agile methodologies. This integration has significantly contributed to elevating the quality of software development.

Traditional methodologies like Waterfall and Spiral, initially employed for structured development, are evolving to align with new technologies, methodologies, and development processes. The integration of these diverse processes does not seek to replace Waterfall or Spiral methodologies but rather to adapt them to a contemporary web environment, enhancing overall performance.

Table 1. Project Inventory

Project/Task	Estimated Proxy Size	Planred Added & Modified Size	Actual Added Modi-field Size	Estimated Hours	Actual Hours
/EBENEZER/Modulo Proveedor_pago	117	147	NA	3.65	4.1
/ITSPTUTORIAS/MODULOS_AGREGAR	94.4	121	105	2.98	2.2
/CURSO PSP/Modulo Usuarios	240	259	215	7.13	5.75
/ELECTRONICA /Modulo Trapaso Inven-tarios/ PSP3	120	131	312	5.55	7.03
/ELECTRONICA /Modulo Nota Servicio	135	191	99	6.12	8.55
/ELECTRONICA /Modulo Corte de caja	314	346	404	14.8	13.7
/ELECTRONICA /Modulo Ventas	387	478	463	15.2	10.1
/ELECTRONICA /Modulo Gastos	215	264	263	9.72	5.07
/ELECTRONICA /Modulo Principal	193	233	194	8.72	7.95
/ELECTRONICA /Modulo Reportes de Inventario	167	207	457	6.13	6.18
/C-LAB/Filtro de Busqueda Cl	253	321	296	9.8	3.12
/C-LAB/Cl inventario	292	361	252	2	2.98
/ELECTRONICA /Modulo Usuarios	163	180	144	5.7	4.42
/ELECTRONICA /Modulo Pedidos	84.7	89.4	94	2.56	2.53
/ELECTRONICA /Modulo Reportes de Caja	240	307	250	8.6	3.28
/EBENEZER/Menu Principal	81	84.8	109	2.46	2.57
/EBENEZER/Modulo Producto	278	325	446	8.45	7.97
/ELECTRONICA /Modulo Bancos	254	316	425	8.0	9.3
/ELECTRONICA /Modulo Compras/ PSP3	236	309	322	10	14.1
/ELECTRONICA /Modulo Gastos Financieros	196	221	184	5.84	6.15
/ELECTRONICA/ Modulo Cuentas Bancar-ias	131	163	60	3	NA
/CURSO PSP/Usuarios	201	219	230	6.26	3.85
/ELECTRONICA /Modulo Proveedor	168	221	328	4.78	8.1
/ELECTRONICA /Modulo ContarClic	48.5	87.5	35	1.76	1.5
/ITSTUTORIAS/MODULOS	90	113	181	2.92	3.08

Table 2. Load/Size Durations

Method	Estimate	r^2	*Beta0*	*Beta1*	Range(70%)	LPI	UPI	Variance	StdDev
B	206	0.53	22.6	1.0	95.6	110	301	7785	88.2
A	237	0.5	23.9	1.16	98.0	139	335	8174	90.4
C2	204		0	1.11					
C1	240		0	1.31					

Table 3. Anticipated Execution Times

Method	Estimate	r^2	*Beta0*	*1/Beta1*	Range(70%)	LPI	UPI	Variance	StdDev
C2	5.63		0	32.6 LOC/Hr					
C1	6.61		0	27.7 LOC/Hr					
C3	5.2		0	35.2 LOC/Hr					
A	6.51	0.4	1.15	34.2 LOC/Hr	3.03	3.48	9.54	7.8	2.79
B	5.72	0.4	1.26	41.1 LOC/Hr	3.04	2.68	8.76	7.86	2.8

Table 4. Proposed Phases

Process Version	PSP0	PSP0.1	PSP1	PSP1.1	PSP2	PSP2.1
Process Scripts and Summaries						
PROBE Estimating Script			x	x	x	x
Forms, Templates, Standards, and Instructions						
Time Recorging Log	x	x	x	x	x	x
Defect Recording Log	x	x	x	x	x	x
Defect Type Standard	x	x	x	x	x	x
PIP		x	x	x	x	x
Coding Standard		x	x	x	x	x
Test Report Template			x	x	x	x
Size Estimating Template			x	x	x	x
Task Planning Template				x	x	x
Schedule Planning Template				x	x	x
Design Review Checklist					x	x
Code Review Checklist					x	x
Use Case Specification Template						x
Functional Specification Template						x
State Specification Template						x
Logic Specification Template						x

Table 5. Proposed Phases

Stage	PSP 2.1 Standard	Web Development
Requirements	Documented requirements statement	
		Sketchboard web
		ER-Diagram
Planning	Project Plan Summary form	
	Size Estimating template	
	Task and Schedule Planning templates	
		Program conceptual design
	Estimated defect data	
	Time and size prediction intervals (PROBE)	
Design	Functional Specification	
	Operational Specification	
	Use Case Diagram and specification	
	Sequence Diagram	
		StateCharat Diagram (se omite para diseños web)
		Relational Diagram Complete
		Dictionary Data
		Data Base Design
		Screenshots of Web View design
Review Design	Design Review Checklist	
		User Review
		Modify to Web View
		Acceptance User

Table 6. Proposed Phases

Code	Coding Standard Review	
	Code	
		Unit Test of Coder
Code Review	Code Review Checklist	
		Coding Standard Verifing
		Unit Test of Coder
Compile		Server Config
		Sources synchronization
		DataBase synchronization
Test	Test Report Template	
		integral Test of User
	Test Result	
Postmortem	Time Recording Log	
	Defect Recording Log	
	Defect Type Standard	
	PIP	
	Size Estimating Complete	
	Schedule Planning Complete	
	Task Planning Complete	

CONCLUSION

Through the progression of each project, a noticeable enhancement in performance was observed by amplifying the identification of defects in the Design and Design Review stages. Simultaneously, productivity experienced an upswing by augmenting the number of lines of code produced per hour, while the cost of quality witnessed a decrease by curtailing the number of defects extending into and beyond the testing phase.

In summary, the introduction of these new and refined stages within the PSP phase represents a clear adaptation to the Web Software Development process. This adaptation instigates fundamental enhancements across modular projects, fostering the overall quality of the software process and the final product.

It is advisable that individuals aspiring to adopt these processes have prior experience with PSP and some familiarity with Web technology. This methodology is tailored for Web Software Development and is not intended to serve as a guide for learning website development languages.

BIBLIOGRAPHICAL REFERENCES

1. Chavarria, A. E., Ore, S. B., Pastor, C. (2016). Quality assurance in the software development process using CMMI, TSP and PSP. Revista Iberica de Sistemas e Tecnologias de Informacao, Vol. 20, pp. 62–77. DOI: 10.17013/risti.20.62-77.
2. Deacon, J. (2000). Model-view-controller (MVC) architecture. John Deacon Computer Systems Development, Consulting & Training.
3. Hilburn, T. B. (1999). PSP metrics in support of software engineering education. Proceedings 12th Conference on Software Engineering Education and Training (Cat. No.PR00131), pp. 135–136. DOI: 10.1109/ CSEE.1999.755194.
4. Hollocker, C. P. (1986). Finding the cost of software quality. IEEE Transactions on Engineering Management, Vol. EM–33, No. 4, pp. 223–228. DOI: 10.1109/TEM.1986.6447683.
5. Humphrey, W. S. (1989). Managing the Software Process. Addison-Wesley Longman Publishing Co., Inc.

6. Humphrey, W. S. (1994). The personal process in software engineering. Proceedings of the Third International Conference on the Software Process. Applying the Software Process, pp. 69–77. DOI: 10.1109/SPCON. 1994.344422.

7. Humphrey, W. S., Over, J. W. (1997). The personal software process (PSP) a full-day tutorial. Proceedings of the (19th) International Conference on Software Engineering, pp. 645–646.

8. Isakowitz, T., Stohr, E., Iyer, B. (1995). RMM: A methodology for structured hypermedia design. ACM, Vol. 38, No. 8, pp. 34–44. DOI: 10.1145/208344.208346.

9. Kemo, S. (2020). Global digital overview. We are social. Accessed 2020-2-2.

10. Markovtsev, V., Long, W. (2018). Public Git archive: A big code dataset for all. Proceedings of the 15th International Conference on Mining Software Repositories, pp. 34–37. DOI: 10.1145/3196398.3196464.

11. Riddle, W., Williams, L. (1986). Modelling software development in the large. 3rd ISPW, IEEE Computer Society, pp. 81–84.

12. Schwabe, D., Rossi, G., Barbosa, S. D. (1996). Systematic hypermedia application design with OOHDM. Proceedings of the the Seventh ACM Conference on Hypertext, pp. 116–128.

13. Suranto, B. (2014). PSP and PQI: How do they improve individual software process. Teknoin, Vol. 20, No. 4. DOI: 10.20885/teknoin. vol20.iss4. art1.