ISSN 2007-9737

# Adaptation of Models and Processes for Web-based Development

Melquizedec Moo-Medina[1], Luis Alberto Muñoz-Ubando[2], Rodrigo Mazún Cruz[1]

[1] SEP/Tecnológico Nacional de México (TecNM),
Instituto Tecnológico Superior Progreso, Yucatán,
Mexico

[2] Tecnologico de Monterrey,
School of Engineering and Sciences,
Mexico

{mmoo,rmazun}@itsprogreso.edu.mx, amunoz@tec.mx

**Abstract.** Currently, technology is a challenge which is rapidly changing the Software Development process, and it is also evolving within the field of Software Engineering itself. In this article, the authors present an adaptation of multiple methodologies and processes in Software Engineering, applied specifically to Web Developments. The Personal Software Process (originally designed by [5]) was developed for the known third generation languages; however, it has been adapted to our current world in such a way that companies are still certified in this process in order to develop software. Throughout twenty-five projects developed using this process combined with (1) Modular Programming, (2) Model View Controller and (3) Agile Methodologies, an adaptation in the process has been achieved in each development. Additionally, the results presented are not only the adaptation to the application in Web Development, but also the incorporation and modification of the activities to carry out to guarantee the quality of the process and the finished product.

**Keywords.** Software engineering, process innovation, custom software process, controller view model, modular programming.

## 1 Introduction

The traditional Waterfall Model [11] has been diversifying through light changes that produce new specific methodologies for these application areas when they are incorporated into new Software Development and Engineering technologies. [5] referring to the *Personal Software Development Process* (**PSP**) mentions: The PSP is a personal process that can be adapted to suit the needs of an individual developer[1].

It is not specific to any programming or design methodology; therefore, it can be used with different methodologies, including Agile software development. Software Engineering methods can be considered to vary from predictive to adaptive. PSP is a predictive methodology, and Agile is considered adaptive, but despite their differences, the TSP / PSP and Agile methodologies share several concepts and approaches, particularly regarding team organization.

In this way, we can see that the process established in PSP can be combined with any other methodology, be it traditional or agile, and additionally they can coexist by adapting their processes to development. In this research, a brand-new empiric and experimentally validated proposal is designed and analyzed to improve and innovate within the PSP methodology. The results are then applied to the development process inside the Web environment through an adaptation to the MVC Layered Model (Model View Controller) [2], using modular programming as an agile methodology.

[1] It became a very popular and easy-going methodology supported by intensive efforts towards its teaching and popularization. It can be understood in a nutshell [7]

ISSN 2007-9737



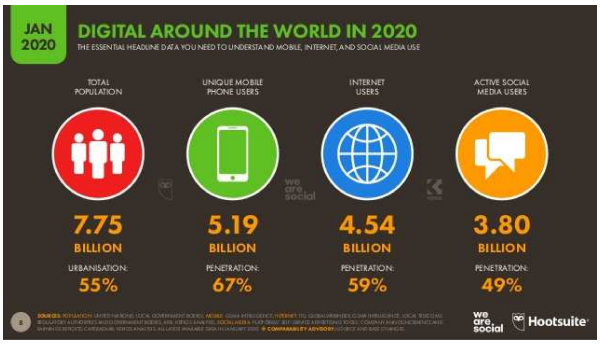


**Fig. 1.** The use of Digital Development

The objective of this research is to present an improvement made to the activities of the PSP combined with the MVC and using modular programming as an agile methodology on twenty-four projects developed in the web environment, through the analysis of performance, productivity and Cost of quality.

The importance of the project stems from the innovation of applying a traditional process based on counting lines of PSP code to a functional, modular process under the MVC that adapts to a main and majority sector such as Web development. As is seen in 1.

The wearesocial studies carried out in January 2020, reveal worldwide penetration levels in terms of internet users, which are 59%. This represents an increase of 14% over the previous year [9].

In the Web software development process, responsive web design is used, so that users of mobile devices can have access to applications. In the previous graph, it is observed that the population of mobile users stands at 67%, with an annual increase of 28%.

The quality of the software is an issue that in itself generates discussion due to differing points of view.

However, in 1986 [5] undertook a project from where he changed the world of Software Engineering, and its final result is a process that allows the measurement of the Quality of Software Development, and therefore, the quality of the finished product.

It is worth thinking about web development through the basic principle, which states: "If there was quality during the Software development process, the finished product will have that quality seal as well." [5].

The advantage of using the proposed methodology is the quality improvement in an incremental and iterative manner; the more it is executed and regulated, the more the process and its deliverables are improved. The times, the sizes, the libraries are enhanced, unleashing a series of benefits which ends in the collection of historical data that will serve for future estimates.

## 2 Problem Statement

The problem to be solved is the lack and limitations of clear and specific methodologies designed for new technologies that are currently emerging. It is such that different frameworks arise adapting to standards designed for the development of desktop software. Although these are used on the Web, it leaves many gaps in knowledge that are considered in desktop software but not in software for the Web. In this way, it can be said that there is not an appropriate software methodology that can provide a solution to these emerging technologies, such as [10], which controls projects as a solution to these needs.

In their struggle to have a better organization during their projects, many Software development companies use a combination of different methodologies to form new ways of working; however, each one manages to adapt these methodologies in a different way.

## 3 State of the Art

This paper describes a position about use of the personal software process (PSP) metrics [5]. The position presented describes how and why PSP metrics can be used in teaching and learning about software engineering. PSP can not solve all problems that students and professionals have in developing software, but it can support and guide said software developers in establishing disciplined practices that can be analyzed and improved upon.

The data and metrics provided by PSP, form the basis for an engineering and scientific approach to such an analysis, and they appear to provide more

promise for success than any other method or tool on the horizon. These metrics also encourage and support a new, more effective paradigm for education that replaces the programming teacher with a coach that is able to give detailed, specific counselling on how students can improve [3].

The personal software process (PSP) provides software engineers a way to improve the quality, predictability, and productivity of their work. It is designed to address the improvement needs of individual engineers and small software organizations. A graduate level PSP course has been taught at six universities and the PSP is being introduced by three industrial software organizations.

The PSP provides a defined sequence of process improvement steps coupled with performance feedback at each step. This helps engineers to understand the quality of their work and to appreciate the effectiveness of the methods they use. Early experience with the PSP shows an average test defect rate improvement of ten times, and an average productivity improvement of 25% or more [6].

Previous works regarding web design methodologies include the RMM (Relationship Management Methodology), OOHDM (Object Oriented Hypermedia Design Method) and UML Based Web, a model based on techniques from object orientation.

— RMM it was originally introduced in [8] and has since evolved in various ways in response to the rapidly growing demand for hypermedia applications on the World Wide Web. The revamped methodology is demonstrated in rich web application design. It discusses Design and implementation issues, including database integration, and top-down and bottom-up approaches toward developing Web Information System (WIS). The graphical and programming language notations for the new RMM constructs are presented. RMM promotes sound design and the sustainable development of hypermedia.

— OOHDM the object-oriented hypermedia design method is a model-based approach to building large hypermedia applications. It has been used to design different types of applications such as: websites and information systems, interactive kiosks, multimedia presentations, among others. It comprises four different activities: Conceptual Design, Navigation Design, Abstract Interface Design and Implementation. They are carried out in a combination of incremental, iterative and prototype-based development styles. Treating interface, navigation and conceptual design as separate activities allows us to focus on different concerns, one at a time. As a result, we get more modular and reusable designs, and we get a framework for reasoning in the design process, encapsulating the design expertise specific to that activity. Furthermore, interface design primitives can be easily mapped to non-object-oriented implementation environments or languages (such as HTML or Toolbook); thus, OOHDM can be used regardless of whether the target system is a pure one, an object-oriented environment, or even a hybrid (like those we usually find on the Internet) [12].

## 4 Methods

The process performs estimates of historical data based on linear regression and average standard deviations.

These were initially calculated for more than twenty-five thousand lines of code, more than sixty-two inserted programs in the database as tests, and correcting fifteen internal versions before its first official version in [5].

We began some methodological changes from the traditional Web Development process towards the MVC methodology. This was done by adapting to the PSP process with modular development. The results show innovative changes to the PSP process applied to Web pages, as well as the solution to problems found for the adaptation of the MVC methodology and Modular development.

To work on the proposal, we have the development of twenty-four modules created under

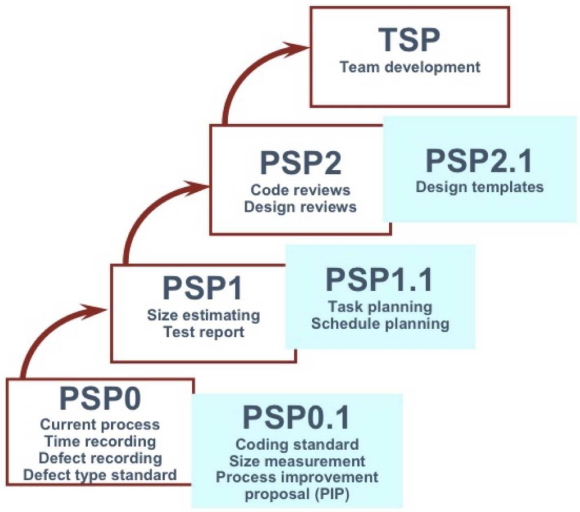


**Fig. 2.** The traditional phases of software development

the Web scope and more than one hundred and seventy delta hours in the PSP process in version 2.1.

### 4.1 Historial Facts

The analysis of the implementation is found in 1, compiled from historical data developed in the methodologies and processes submitted in the initial section of the proposal.

### 4.2 Testing Methods for Size

Taking into account the development history outlined in Table 1, when generating a new module in the project, an example of the estimations of this module is shown to the general project of the system and it is explained how the estimation is carried out.

The calculation of the estimation of code lines to be programmed for a 183 LOC module (Lines of Code) is 206 LOC in the estimation method B which is based on the planning data and 237 LOC in the estimation method A based on historical data. These have a deviation of 88.2 and 90.4 respectively, as shown in 2.

This indicates that method B is the best option to select, due to the precision of the historical data entered in the planning stage that ensures a better estimate in the development of the next module.

### 4.3 Estimated Productivity

The productivity estimate for planned development sizes and times is 42.2 LOC / Hr, which is consistent to the current date with a productivity of 35.2 LOC / Hr. (± 19.5).

### 4.4 The Value Proposition of this Paper

The proposal is based on the modification of the traditional PSP process that was initially developed for third generation languages, inserting key tasks and activities that adjust to this process and adapt it to Web Development.

A key factor will be the correct interpretation of historical data that allow estimations on the scope of the quality of the software. Different methodologies will be used for this, such as the MVC, Modular programming and agile methodologies that bring the user or client closer to the Development process.

The expected outcome will be a new process of Software Development for the Web field which is both innovative and functional.

### 4.5 Proposed Methodology

To start the adaptation to the Development process, you must begin with the application of the PSP 0, 0.1, 1, 1.2, 2 and 2.1 successively, as shown in 2. During the adaptation of the process, changes that have a direct impact have been added to web application development.

In PSP, all the tasks and activities that the software engineer must carry out during the development process of a software product are specifically defined in a set of documents known as scripts. These must be followed in a disciplinary way since the success of the improvement sought out will depend on it. (The Personal Software Process (PSP).

To carry out the proposal, the PSP 0 process is applied first, which consists of developing the software on a daily basis, only increasing the recording of development times on the Waterfall methodology. The data collected will serve for

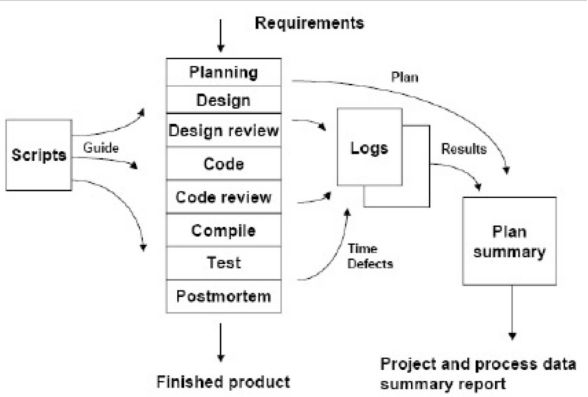


**Fig. 3.** The PSP phases of software development

future estimates. At this stage, it is optimal to create changes in the design stage by adding the Web design (Web View Design) that consists in the generation of the HTML tagging already with the responsive styles of bootstrap and CSS3, or according to the Design methodology.

In the MVC, this change consists of producing, in the project's View folder, the Web interface without functionality and presenting it to the user for approval (User Review).

Before moving on to coding, the changes requested by the user will be modified in the design stage. This single change will serve to generate confidence and concentration in the use of time control.

At the end of the Compile stage, the generated module will be presented to the client again for approval as part of the Test.

The next step is to apply the PSP 0.1 process which allows us to add the code line count in the development process. The objectives of PSP 0.1 are: "to measure the size of the programs that are produced, to perform the size counting for the programs and, to make precise and exact measurements of size".

(Using PSP 0.1) In order to approach the number of functions that will be carried out and get a more accurate count; in this step, it is recommended to both add the draft of the module that will be implemented (Sketchboard Web) in the requirements and obtain the ER diagram with the functionalities in the requirements survey stage.

Likewise, it is suggested to add the unit tests of the Programmer (Unit Test of Coder) in the coding stage. This consists of verifying the functionalities programmed during this stage on localhost.

In PSP 1, the personal planning process is added, which consists of establishing an orderly and repeatable procedure to develop software size estimates. (Using PSP 1, 2006).

At this stage, it is suggested to add the technical conceptual design of the software. This will help demonstrate the relationship of the files that will be used in the development, and assist in organizing the MVC and its functions.

In this case, it can also be adapted to the use of a Development Framework.

The next step of Growth is the use of PSP 1.1 which aims to introduce and practice methods to make resource plans and schedules by tracking performance against these plans, and establishing probable project completion dates.

At this stage, it is suggested to also add to the process the synchronization of the Database and the source files to the server during the compilation stage in a designated section for tests, so that in turn it is validated by the user in the Test stage.

In the PSP 2 stage, two stages presented in the PSP numbering are inserted, which aims to increase the quality of the development by looking for injected defects in the design and coding stages.

At this stage it is suggested to verify the design and coding standards developed in the PSP 0.1 process [13].

In the PSP 2.1 stage, Design combined with UML is added in the Development process.

At this stage, additional measures are introduced to manage the quality of the process, as well as design templates that provide an orderly framework and format for recording designs.

The following list shows the elements added 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,

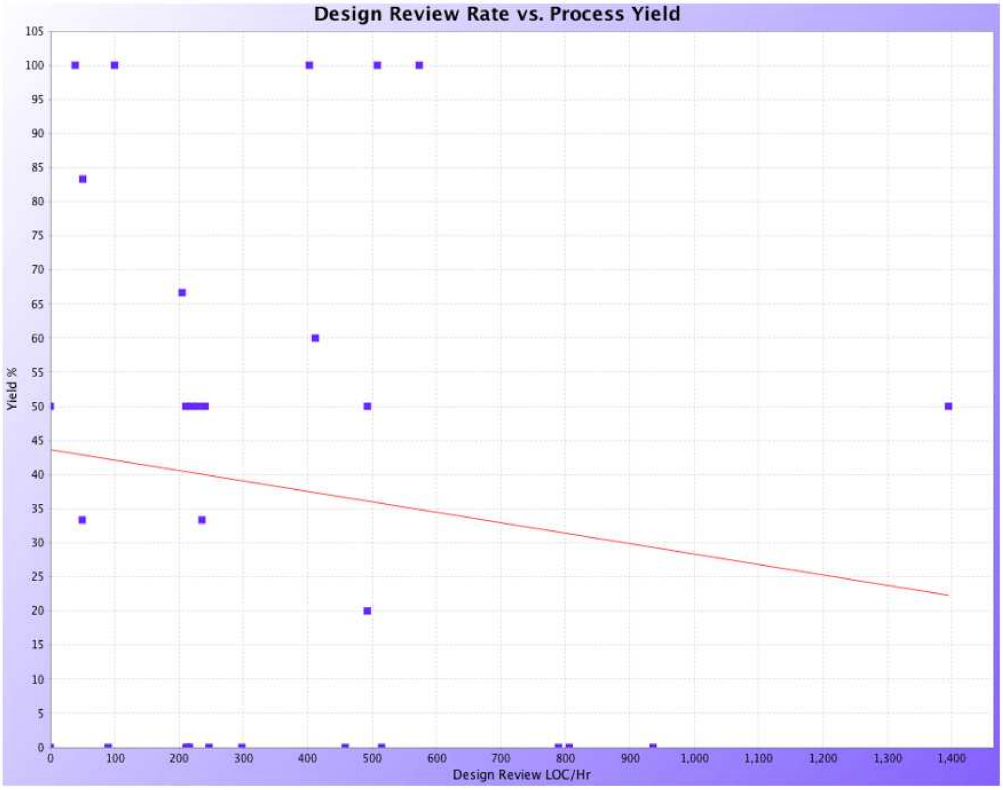


**Fig. 4.** Experimented process

5. State specification template,
6. Logic specification template.

The deliverables that have been made as an improvement in this stage for the Web design are the Use Case Diagram and Specification, the sequence diagram, the State diagram, the Relational Diagram, the Data Dictionary and the Database Design Data.

In summation, the activities and the deliverables, forms, templates or standards requested by PSP in its different versions as described in this section, are shown in 4.

Specifically, in version 2.1, all the deliverables of the other versions can be verified.

Table 4 shows where the deliverables should be prepared according to the PSP version chosen.

In Table 5, it is observed that, starting from the previous table and with the stages already classified, the deliverables added to ensure functionality in Web development and its quality are shown on the right side of the table.

## 5 Experimental Results

The main objectives of PSP are: maximize software quality, achieve a discipline of continuous improvement in the development process, improve the quality of the development process and increase productivity. Considering that quality is based on measurement, PSP integrates a set of forms, which are also used by PSP to collect the necessary metrics [1]. There are three basic metrics that PSP establishes such as size, time, defects, and the others are measurements derived from these basic metrics. Defects is a metric related to software quality and using PSP can reduce defects as stated in [1].

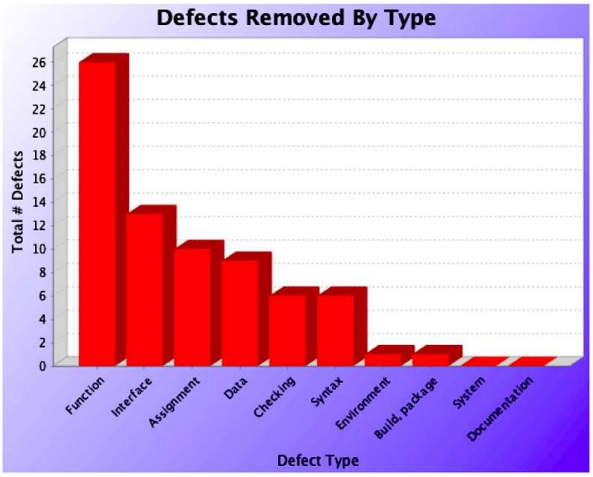


**Fig. 5.** Failure inventory

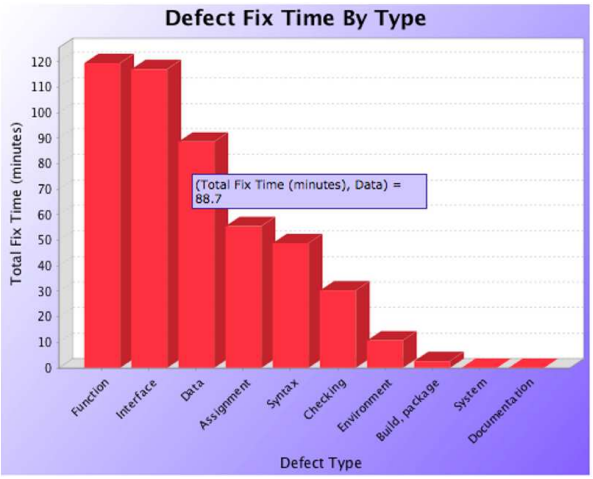


**Fig. 6.** Time lag for failure solving

### 5.1 Performance Results

In the PSP, performance is measured in two ways: the percentage of total defects found and removed in a stage, and Process Yield, which refers to the percentage of defects removed before the first test, compilation and testing 3.

In 4, it is presented the statistics of the Performance obtained in the projects during the design review stage with a negative slope from 45% to 22%, this indicates that the number of defects decreases in the progress of the projects in the Design review, and more defects are repaired before reaching the testing stage.

### 5.2 Results of Defect Classification

The results of the total types of defects injected into the system in the development of twenty-four projects in the Web field are shown in 5 and 6.

In the graph on the left, it can be seen that the number of defects per interface is less than the functional one, while on the graph on the right it can be seen that the time to repair defects in interfaces is as high as the functional one with an approximate time of 120 minutes. This shows that there are few repairs in the Interfaces, but with a sum of minutes equal to the repair time of the functionality.

The question that arises when observing the graphs is: Why does it take longer to make a repair in Interface than in Controllers or Models? This increase is derived from the adaptation of three activities during the design review:

— User Review.

— Modify to Web View.

— Acceptance User.

These three activities are not included in the traditional PSP and are necessary for web development, since they integrate the end user or client in the design review stage.

Here, they accept the Web view or request modifications to be included in the requirements. The Modify to Web View activity is optional only if modifications are required. These elements are a part of agile methodologies.

The advantage of completing these activities is that when coding begins, there is already a design that is accepted and endorsed by the user. Therefore, it would be very difficult to find a design error on something that has already been double validated, resulting in a beneficial performance of 4.

At the same time, this helps reduce possible design defects in the testing stage and, as in Figure 4, performance increases due to the decrease in defects to be repaired during the testing stage.

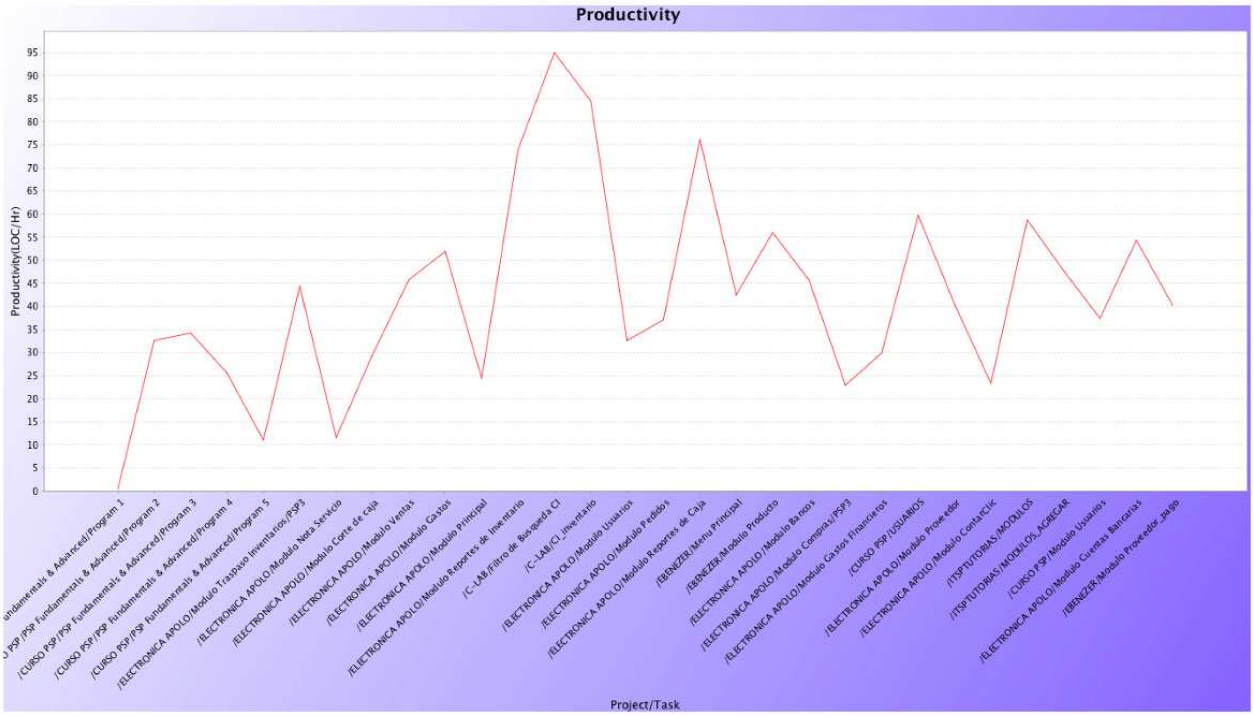


**Fig. 7.** The productivity improvement

Both 4, 5 and 6 support the quality of the software due to the improvement in performance.

### 5.3 Productivity Results

Productivity in PSP is measured by the number of Lines of Code (LOC) developed per hour. In Figure 6, it can be observed how average productivity went from 30 LOC / Hr. in the first program, to 40 LOC / Hr. and has a positive slope.

In the same way, a difference is observed between the less complex modules which raise productivity above 50 LOC / Hr. and more complex ones up to 40 LOC / Hr. Likewise, an exception is observed in the CI search filter module, which had a productivity of close to 90 LOC / Hr. due to a change in user requirements during the design stage.

Productivity allows for a self-evaluation to be carried out each time a project is completed, from which it can be concluded whether there is an improvement in the production process.

### 5.4 Quality Cost Results

The failure rate relationship measures the quality of the engineering process, using quality cost parameters, as stated by the application of Juran's quality improvement program. This is presented to assist software engineering management in the identification and control of quality costs [4]. Evaluation quality cost is the percentage of development time spent on quality evaluation activities.

From project 6, we observed an increase in the cost of quality due to the increase in defects in the testing stages or final stages of development. In the last 10 projects, a slight decrease in the cost of quality was noted due to the decrease in defects in the final stages, which mostly occurs when the user is included in the Review Design stage.

## 6 Discussion

According to the results obtained, it was possible to observe an improvement in the quality of

**Table 1.** The list of projects

| Project/Task | Estimated Proxy Size | Planred Added & Modified Size | Actual Added Modifield 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 Inventarios/ 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 CI | 253 | 321 | 296 | 9.8 | 3.12 |
| /C-LAB/CI 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 Bancarias | 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 times

| 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.** Estimated executing 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 |

development, the process and the results of the process by combining the methodologies of PSP, Modular Programming, MVC and agile methodologies. All of which contributed to increasing the quality of Software Development.

The primary methodologies such as Waterfall and Spiral, were used in principle for structured development and are evolving to adapt to new technologies, methodologies and Development processes. This combination of processes is not intended to replace Waterfall or Spiral methodologies, but take them to an updated Web environment with better performance.

## 7 Conclusion

Throughout the development of each project, it was observed that there is an increment in performance by increasing the number of defects found in the Design and Design Review stages. Additionally,

**Table 4.** Phases proposed

| 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.** Phases proposed

| 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.** Phases proposed

| | | |
|---|---|---|
| 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 | |

productivity was increased by increasing the number of lines of code written per hour, and the cost of quality was reduced by minimizing the number of defects that make it to and beyond testing.

It can therefore be concluded that establishing these new and refined stages during the PSP phase presents a clear adaptation to the Web Software Development process.

This contributes to setting up basic improvements throughout the application in the modular projects, and promotes the quality of the software process and the finished product.

It is recommended that those who wish to join these processes have previously used PSP and some Web technology. Because this process is a Web Software Development methodology, and is not intended to assist in learning Website development languages.

## 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.

*Article received on 29/04/2021; accepted on 27/08/2022.*
*Corresponding author is Melquizedec Moo-Medina.*