Revista Cubana de Ciencias Informáticas
Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 84-101

Tipo de artículo: Artículo original
Temática: Ingeniería y gestión de software
Recibido: 21/06/2022 | Aceptado: 01/07/2022

Procedure for carrying out portability tests at the University of Informatics Sciences

Procedimiento para realizar las pruebas de portabilidad en la Universidad de Ciencias Informáticas

Leosmay Carrión Estradet [1] https://orcid.org/0000-0003-3581-156X
Amanda Delgado Martínez[1] https://orcid.org/0000-0002-0349-6174
Maidelyn Piñero González[1] https://orcid.org/0000-0002-6534-132X
Aymara Marin Diaz[1] https://orcid.org/0000-0001-5101-7804

[1]University of Computer Science, Cuba. Highway San Antonio Km 2 1/2. {lcarrion, adelgado, mpinero, amarin}@uci.cu

* Author for correspondence. (lcarrion.uci.cu)

ABSTRACT

The fundamental factor for success in software production is quality and for this it is necessary to take into account a series of aspects so that it is optimal. Obtaining quality software implies the use of standard procedures for the analysis, design, programming and testing of the software, which allow standardizing the work philosophy in order to achieve greater reliability, maintainability and ease of testing, while increase productivity for both development work and software quality control. Before the software is delivered to the end user, it is necessary to carry out tests in order to detect errors in the application and the documentation; this process is of great importance since it gives a measure of the quality of the product as long as it is

carried out appropriately. The main objective of this work is to propose a procedure to carry out portability tests on software products, which will explain the activities and tools that will be used. The application of this procedure will contribute to improving the quality of the systems, revealing nonconformities that are difficult to detect through functional tests.

Keywords: procedure; test; quality; defects; software; portability.

RESUMEN

El factor fundamental para el éxito en la producción de software es la calidad y para ello es necesario tener en cuenta una serie de aspectos para que la misma sea óptima. La obtención de un software con calidad implica la utilización de procedimientos estándares para el análisis, diseño, programación y prueba del software, que permitan uniformar la filosofía de trabajo en aras de lograr una mayor confiabilidad, mantenibilidad y facilidad de prueba, a la vez que eleven la productividad tanto para la labor de desarrollo como para el control de la calidad del software. Antes de que el software se le entregue al usuario final es necesario realizar pruebas con el objetivo de detectar errores de la aplicación y la documentación; este proceso resulta de gran importancia, ya que da una medida de la calidad del producto siempre que se lleve a cabo de forma apropiada. El objetivo principal de este trabajo es proponer un procedimiento para realizar pruebas de portabilidad a productos de software, el cual explicará las actividades y herramientas que se emplearán. La aplicación de este procedimiento contribuirá a mejorar la calidad de los sistemas, revelando no conformidades difícilmente detectadas mediante pruebas funcionales.

Palabras clave: procedimiento; prueba; calidad; defectos; software; portabilidad.

Introduction

The development of the software industry from its beginnings to the present has evolved considerably, giving rise, over the years, to the emergence of increasingly complex and varied software. Today almost everyone can develop software, from self-taught fifteen-year-olds to large production companies; and the products can range from simple games for children, educational multimedia and management software, to air transportation systems, medical, etc. All this boom brought with its competition and increasingly demanding customers. That is why the quality of the product reaches a significant value for both customers and producers. With so many options, only the best products will manage to take a place of renown. (ISO9000, 2000, Castillo, Mora et al., 2017, Castillo, Mora et al., 2018, A. I. Vlasov, 2022)

Software Quality Management is a fundamental pillar throughout the life of a product, with the purpose of understanding customer expectations in terms of quality, and implementing a proactive plan to meet those expectations. The software testing procedure is an essential and critical mechanism for the validation of a product. Currently, software development, due to its multiple applications in business, has become one of the fundamental elements of information technology and communications, bringing with it the development of systems in all sectors of society that support the different processes that understand. (Carrizo and Alfaro, 2018, Aizprua, Ortega et al., 2019, Normalización, 2019)

The system testing application is responsible for checking the proper functioning and quality of the software, both functional requirements and non-functional requirements. There are several types of system tests that are applied to software such as: security tests, performance tests, portability tests, among others (Bibián, 2017, Cárdenas Hernández, 2019). To control this process, it is necessary to use guides, which, supported by these tests, it is possible to define a strategy both with functional tests with the corresponding test cases according to the specifications of the requirements and for non-functional tests, which contain the non-functional requirements. Described in the application and in case of the existence of a requirement, a test case is made for this type of tests. (Verona-Marcos, Pérez-Díaz et al., 2016, Atoum, Baklizi et al., 2021)

At present, the evaluations of the non-functional requirements of the software are insufficient, since they are subjective characteristics that are complex to measure. The development and testing of the required

functionalities, meeting delivery times and minimizing costs are prioritized, leaving aside the evaluation of non-functional requirements (Diaz, Casañola et al., 2018). This brings with it that nonconformities are detected at runtime that could have been found in a controlled testing environment, increasing the time, effort and costs in their resolution (Diaz, Casañola et al., 2018, Diaz, Casañola et al., 2020, González, Diaz et al., 2021).

Ignorance of the behavior of the software on different types of devices generates discontent in users who do not have a mastery in system navigation. Sometimes, when updating the applications, users must relearn how to use it, since the interfaces are very different, as are the functionalities. An interview with 30 users between 14 and 40 years of age who use national mobile applications, showed that there is difficulty in understanding the applications when they are updated, it is cumbersome to adapt to the new operation and interface, since 100% of respondents mention it.

On the other hand, a survey of 25 specialists from different projects at the University of Informatics Sciences (UCI) identified that for 100% the use of portability tests is important for the acceptance of the product by users, however, 83% do not take them into account in the project to which they belong. An analysis carried out by the specialists, testers and senior management of the Software Quality Department at the UCI revealed the need to incorporate the evaluation of this characteristic through portability tests. Due to the above, it is necessary to know the behavior of the software portability during its development, for when it is deployed, there is evidence of improvement in understanding and compression by end users. Responding to this need, it is proposed to carry out a procedure to perform portability tests on any software product in the UCI.

Computational Methods or Methodology

Several scientific methods were used to obtain research information. Among the theoretical methods the historical-logical was used for the critical study of the previous works, and to use these as a point of reference and verification of the results achieved; the inductive-deductive to reach conclusions about the

research problem, from the generalization and specification of the partial results that are obtained, the analytical-synthetic for the analysis of the bibliography about the most used quality models internationally and the deductive hypothetical for the identification of the problematic situation and the solutions. Within the empirical methods, the interview was used to obtain information on the performance of users with the product under evaluation for portability; In addition, a survey was carried out to collect information and obtain the criteria of the experts in guaranteeing the quality of the software.

Generalities

Software quality characteristics have been defined by product-level quality standards and norms in three types: internal, external, and in-use. This approach is oriented to verify the quality of the software product to achieve the satisfaction of the client or end user regarding the requirements associated with portability in the initial stages of the software development process. ISO/IEC 25010:2011 defines portability as the degree of effectiveness and efficiency with which a system, product or component can be transferred from one hardware, software or environment (operational or use) to another. Set the following sub-features for this feature (ISO/IEC, 2011, Hovorushchenko and Pomorova, 2016, Paz, Gómez et al., 2017, Yuan, Salgado et al., 2020):

1. Adaptability: degree to which a product or system can be adapted effectively and efficiently to different evolving hardware or software, or other usage or operating environments.
2. Instability: degree of effectiveness and efficiency with which a product or system can be installed and/or successfully uninstalled in a specific environment.
3. Replaceability: The degree to which a product can replace another specific software product for the same purpose in the same environment.

To evaluate the portability characteristic are defined as basic requirements to be met by the UCI products, which are shown below. See Table 1.

Table 1 - General requirements for the portability feature. (CALISOFT, 2017)

Subfeature	Defined requirements
Installability	The actual installation time must be less than or equal to the expected time. The client must submit the Configuration Sheet with the required data, in which the configurations in which the test will be executed are found.
Replaceability	The user functions of the replaced product should be performed without any additional training or workaround. NOTE: User functions are those that the user can call and use to perform the intended tasks, including user interfaces. The behavior of the quality measures of the new product must be better than or equal to the replaced product. The number of functions that produce similar results should be easily used after replacing the old software product with the current one. It should be possible to import the same data after replacing the old software product with the current one.
Adaptability	The software or system must be capable enough to adapt to different hardware, software or other operating environments. The software operation test should be performed from the maintenance point.

Procedure Description

The software testing procedure integrates a set of activities that describe the steps that must be carried out in a testing process such as: planning, design of test cases, execution and results, taking into account how much effort and resources will be required, in order to obtain a correct software construction as a result. Within the procedure, it is necessary to take into account the human resources that must intervene and each one of them must know their responsibilities, in order for the process to be well executed. Next, the roles and responsibilities that will be involved in the procedure to carry out the portability tests at the university are defined. See Table 2.

Table 2 - Roles and Responsibilities

Role	Responsibilities
Head of	Review test requests.

Department	Assign requests to test coordinators. Call startup meetings. Approve the Test Plan. Evaluate the testing process Close the testing process
test coordinator	Create in the repository the Test File of each request. Weekly update the status of the tests for each request. Prepare the Pre-Test Plan. Lead the Start of the Release Test Process Meeting. Design the test. Guide the development of tests and keep those involved informed. Ensure compliance with the Test Plan approved at the kick-off meeting. Send the defect report to the development team at the end of each iteration. Evaluate testers at each test iteration. Reconcile the defects declared Not Applicable with the Project Manager, at the beginning of each iteration. Keep the test file updated in the repository. Lead the Closing Meeting.
Test Advisor	Run Sampling Evaluate the testing process
Technological Advisor	Manage server virtualization with Configuration Manager. Prepare the test environment. Once the tests are finished, eliminate the environment.
Quality Advisor	Upload to gespro.dgp.prod.uci.cu the Release Test application. Participate in the Initial Meeting of the process. Check that the project is complying with the Test Plan. Participate in the closing meeting of the tests.
Configuration Manager	Virtualize the servers. Once the tests are finished, delete the environment.
Project manager	Make the request for Release Test. Participate in the Initial Meeting of the process. Reviews and approves the artifacts generated during testing. Ensure compliance with the times established for the resolution of defects.

	Guarantee the response to all defects. Reconcile detected defects with the Test Coordinator Participate in the closing meeting of the tests.
Testers	Execute the tests from the description of the test cases and the supporting artifacts. Record and classify detected defects. Send the Individual Defect Report to the test coordinator.
Clients, Senior Management	Decision making.
Development team	Participate in the Initial Meeting of the process. Participate in each test iteration to clarify any doubts about the business of the application. Respond to the defects detected in each iteration, explaining, if applicable, why none of them proceed. Participate in the closing meeting of the tests. Follow-up of the execution of the tests. Resolution of detected nonconformities.

Font: Own elaboration.

Procedure

The proposed procedure to carry out the portability tests in the ICU consists of 4 stages: Planning, Design, Execution and Analysis of results. It is shown below in Fig. 1:

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 84-101

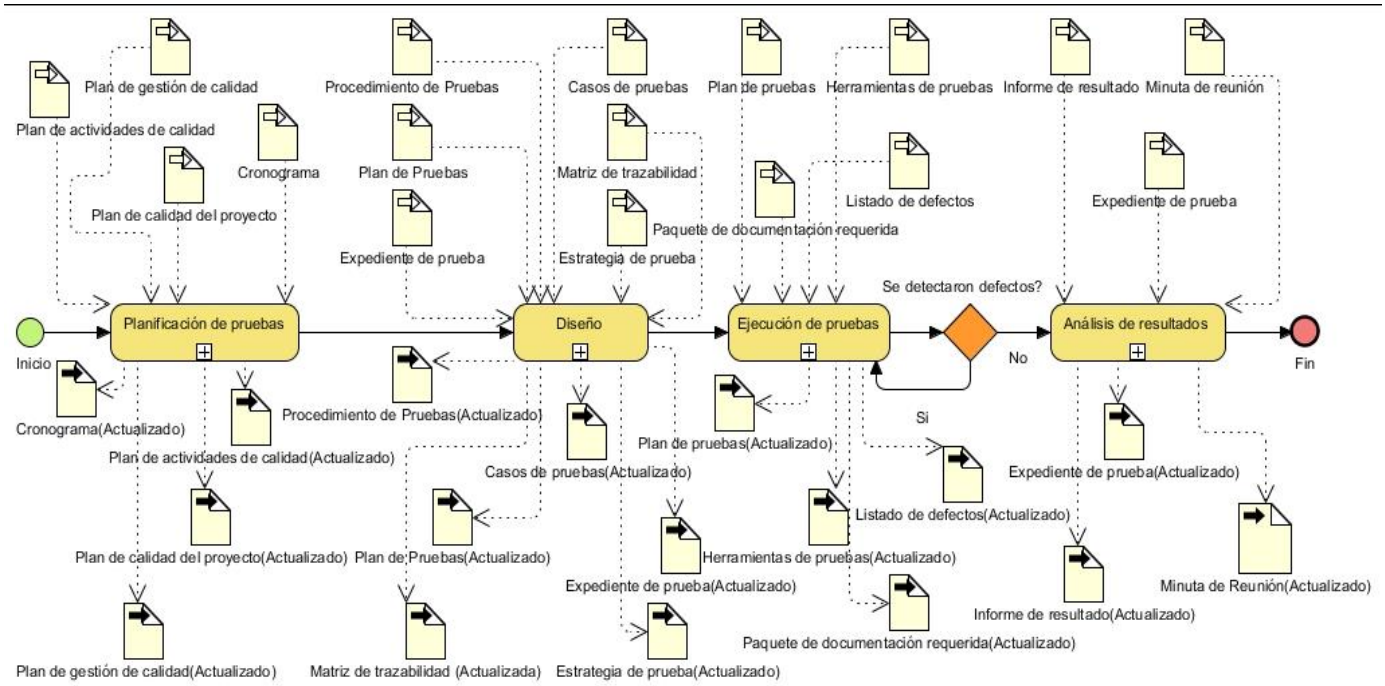

Fig. 1- Software portability process.

Font: Own elaboration.

Portability tests are planned initially by the Quality Advisor of the Development Center who makes the test request, which will be accepted or rejected by the Laboratory Quality Coordinator once the review of the test file is completed. If the request is rejected, the Quality Advisor will analyze the causes and carry out the corresponding actions. If accepted, it is analyzed by the head of the Laboratory who will check the availability of resources, the workload of the coordinators and testers to assign the activity. Then, the Responsible Coordinator summons the project specialists to the corresponding kick-off meeting, always three days before the start date of the evaluation process.

The following activities are carried out in **planning:**

1. Initial planning according to the project schedule.
2. If there are no deviations in the quality management plan, the same planning is maintained and if any deviation occurs, the quality management plan is updated.

3. The quality advisor is notified by mail once the project's Quality Plan is updated.
4. The Quality Activities Plan of the Quality Department is updated.

Within the **Design stage,** the following activities are carried out:

1. Analysis of defined portability requirements.
2. Tools are identified by test type.
3. The testing strategy is defined.
4. A design of the checklists is made.
5. Submitted test cases are reviewed.
6. Update test cases.
7. Prepare automated tests.
8. Carry out the test traceability matrix.
9. The test strategy is updated.
10. Update evidence file.

Within the **Execution stage,** the following activities are carried out:

1. Create the test environment.
2. Run iteration.
3. Record defects/anomalies in the Defect Log.
4. Check defects.
5. Notify the result.
6. Respond defects.
7. Reconcile defects in the case that they do not proceed.
8. Run regression test in case defects proceed.
9. Update the documentation with the list of approved defects.

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 84-101

Then, in the **Results Analysis stage**, an analysis of the tests carried out is carried out to find out if the product has software portability or whether or not the system meets the previously defined requirements. The following activities are carried out:

1. Analysis of defects found: number of defects, impact, work product affected, as well as the cause of the defect.
2. Make conclusions and recommendations.
3. Update project file
4. Prepare release certificate.
5. Update the test file.

Metrics to measure the coverage of the tests

The quality measures to be taken into account to evaluate portability are based on the ISO/IEC 25023:2016 Standard, since it establishes it for all software and taking into account each sub-characteristic. They are shown below:

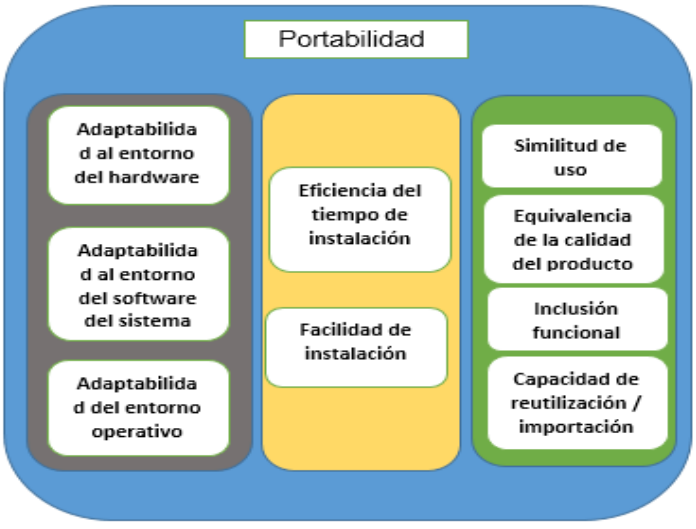

Fig. 2 - Quality measures for the portability feature**.**

Source: (ISO/IEC 2016)**.**

Revista Cubana de Ciencias Informáticas
Vol. 16, No. 3, Julio-Septiembre, 2022
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 84-101

Instability measure

Table 4 - Installability measures. (ISO/IEC, 2016)

ID	Name	Description	Measurement function
PIN-1-G	Installation time efficiency	How efficient is the actual installation time compared to the expected time?	A_i= Total working time dedicated to installation i B_i= Expected time to perform an installation i n = Number of installations measured
NOTE 1: X greater than 1 represents an inefficient installation, and X less than 1 represents a very efficient installation.			
PIN-2-G	Ease of Installation	Can users or maintainers customize the installation procedure for their convenience?	$X = 1 - A/B$ A = Number of cases in which a user manages to customize the installation procedure B = Number of cases in which a user attempted to customize the installation procedure for the user's convenience
NOTE: These installation procedure changes may be recognized as customization of the installation by the user.			

Replacement measures

Replaceability measures are used to assess the degree to which a product can replace another software product specified for the same purpose in the same environment. See Table 5.

Table 5 - Measures of replaceability.(ISO/IEC, 2016)

ID	Name	Description	Measurement function
PRe-1-G	similarity of use	What proportion of user functions of the superseded product can be performed without any additional training or workaround?	$X = A/B$ A = Number of user functions that can be performed without any additional learning or workaround B = Number of user functions in the

ID	Name	Description	Measurement function
			superseded software product
NOTE: User functions are those that the user can call and use to perform the intended tasks, including user interfaces.			
PRe-2-S	Product quality equivalence	What proportion of the quality measures is satisfied after replacing the old software product with this one?	$X = A/B$ A = Number of quality measures of the new product that are better than or equal to the replaced product B = Number of quality measures of the replaced software product that are relevant
NOTE: Some of the product qualities that are relevant to interchangeability are interoperability, security, and performance efficiency.			
PRe-3-S	functional inclusion	Can similar functions be easily used after replacing the previous software product with this one?	$X = A/B$ A = Number of functions that produce results similar to the previous ones B = Number of features to be used in the replaced software product
PRe-4-S	Reusability / data import	Can the same data be used after replacing the old software product with this one?	$X = A/B$ A = Number of data that can be used continuously as before B = Number of data to be used continuously in the superseded software product

Adaptability measures

Adaptability measures are used to assess the degree to which a product or system can be effectively and efficiently adapted to different or evolving hardware, software, or other operating environments of use. See Table 6.

Table 6 - Measures of adaptability.(ISO/IEC, 2016)

ID	Name	Description	Measurement function
PAd-1-G	Adaptability to the hardware environment	Is the software or system capable enough to adapt to	$X = 1 - A/B$ A = Number of functions that were not completed or

ID	Name	Description	Measurement function
		different hardware environments?	insufficient results to meet the requirements during the tests B = Number of features that have been tested in different hardware environments
PAd-2-G	Adaptability to system software environment	Is the software or system capable enough to adapt to different system software environments?	$X = 1 - A/B$ A = Number of functions that were not completed or insufficient results to meet the requirements during the tests B = Number of features that have been tested in different system software environments
NOTE 1 When a user has to apply a fitting procedure that has not been previously provided by the software for a specific fitting need, the user effort required to fit should be measured. NOTE 2: System software may include operating systems, midelware (communication channel between software and hardware), database management system, compiler, network management system, etc.			
PAd-3-S	Adaptability of the operating environment	How easily can the test run be carried out from the maintenance point?	$X = 1 - A/B$ A = Number of functions that were not completed or insufficient results to meet the requirements during the operational tests with the user environment B = Number of functions that have been tested in different operational environments

Validation of the proposal

For the evaluation of the given proposal, it was necessary to carry out a survey that would allow obtaining expert criteria. For the selection of these, the curricular analysis technique was applied. When selecting the experts, the following initial selection criteria were considered: knowledge related to software quality, practical experience of activities associated with portability in software projects, work experience in the software industry of 6 years or more. And research associated with the object of investigation. Taking these criteria into account, a questionnaire for curricular knowledge was carried out and as a result 8 experts from the institution were chosen.

Then the Delphi method was applied, 2 rounds were carried out, the answers given were analyzed and specified. The results issued were satisfactory, since all the categories were evaluated as Very High or High, validating the contribution of the procedure proposal to carry out portability tests in software development. In addition, it was reaffirmed as a necessary feature to take into account in the software and essential for the satisfaction and acceptance of the product by end users. A mode of High or Very High was obtained and the experts did not cast votes on the scale of Low or None. From the votes cast by the experts, the following results are obtained. See Table 7.

Table 7 - Percentage of the expert's criteria.

Criterion	Percent	Fashion
Relevance	93	5
Relevance	95	5
Coherence	78	4
Comprehension	88	4
Accuracy	75	4

Based on these previous results, it can be ensured that the experts consider that the procedure proposed to carry out the portability tests allows the detection of software defects associated with this characteristic. Thus, it makes it possible to improve the comprehension and understanding of the systems by end users. It is recognized that it is a fundamental characteristic for the satisfaction of end users.

Conclusions

After conducting the investigation, it is concluded that:

1. The definition of a procedure to carry out the portability tests in the ICU software defines the sequence of activities to be carried out to validate this characteristic in the institution's applications.
2. The roles and responsibilities that will intervene during this process were defined, which makes it

possible to identify the necessary resources and the skills of each role.

3. The quality measures to evaluate this characteristic are those defined by the ISO/IEC 25023:2016 Standard.
4. The satisfactory results of the validation showed that the proposal allows to identify the defects associated with portability and a better understanding of the systems by the end users.

References

A. I. Vlasov, B. V. A., and L. V. Juravleva (2022). "Quality estimation method in advanced software systems." AIP Conference Proceedings 2467.

Aizprua, S., A. Ortega and L. V. Chong (2019). "Calidad del Software una Perspectiva Continua." Revista científica CENTROS 8: pp.120-134.

Atoum, I., M. Baklizi, I. Alsmadi, A. A. Otoom, T. Alhersh, J. Ababneh, J. Almalki and S. Alshahrani (2021). "Challenges of Software Requirements Quality Assurance and Validation: A Systematic Literature Review." IEEE Access.

Bibián, O. P. J. (2017). PRUEBAS DE CALIDAD APLICADAS AL SITIO WEB ALLISON. Máster OBTENER EL GRADO DE MAESTRO EN CIENCIAS COMPUTACIONALES, Tecnológico Nacional de México, Instituto Tecnologico de Colima.

CALISOFT (2017). REQUISITOS DE LA CALIDAD PARA SISTEMAS INFORMÁTICOS Y PRODUCTOS DE SOFTWARE. CALISOFT. La Habana, Cuba, 03/2017: 12.

Cárdenas Hernández, W. A. (2019). Elaboración de un marco de trabajo para pruebas de software, basado en el estándar ISO/IEC/IEEE 29119 y su impacto en el proceso de evaluación del software.

Carrizo, D. and A. Alfaro (2018). "Método de aseguramiento de la calidad en una metodología de desarrollo de software: un enfoque práctico." Revista Chilena de Ingeniería Vol. 26 Nº 1: pp. 114-129.

Castillo, F. F. R., N. M. L. Mora, K. D. C. Elizaldes and J. I. P. Orozco (2017). "ESTADO DEL ARTE: MÉTRICAS DE CALIDAD PARA EL DESARROLLO DE APLICACIONES WEB." 3C Tecnología Vol.6 – Nº 4.

Castillo, F. F. R., N. M. L. Mora, K. D. C. Elizaldes and J. I. P. J. c. T. g. d. i. a. a. l. p. Orozco (2018). "Comparación de métricas de calidad para el desarrollo de aplicaciones web." 7(3): 94-113.

Diaz, A. M., Y. T. Casañola and D. B. Hidalgo (2018). "Marco de Trabajo para gestionar actividades de calidad." 12, No. 2.

Diaz, A. M., Y. T. Casañola and D. B. Hidalgo (2020). "Estrategia de pruebas para organizaciones desarrolladoras de software." Revista Cubana de Ciencias Informáticas Vol. 14, No. 3: 22.

González, M. P., A. M. Diaz, Y. T. Casañola and D. B. Hidalgo (2021). "Buenas prácticas para prevenir los riesgos de la eficiencia del desempeño en los productos de software." Revista Cubana de Ciencias Informáticas Vol. 15, No. 1: Pág. 89-113.

Hovorushchenko, T. and O. Pomorova (2016). "Evaluation of Mutual Influences of Software Quality Characteristics Based ISO 25010:2011." CSIT 2016 80: 4.

ISO9000, N. I. (2000). Sistemas de gestión de la calidad — Fundamentos y vocabulario. Ginebra, Suiza: 42.

ISO/IEC (2011). ISO/IEC 25010:2011 Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models, Switzerland: 44.

ISO/IEC (2016). ESTÁNDAR INTERNACIONAL ISO/IEC 25023 Sistemas e ingeniería de software Calidad de sistemas y software Requisitos y Evaluación (SQuaRE) - Medición de la calidad del producto del sistema y software. Switzerland: 61.

Normalización, O. N. d. (2019). NORMA CUBANA ISO/IEC/IEEE 29119-1:2019 INGENIERÍA DE SOFTWARE Y SISTEMAS – PRUEBAS DE SOFTWARE – PARTE 1: CONCEPTOS Y DEFINICIONES. ISO/IEC/IEEE 29119-1:2013, IDT, Cuban National Bureau of Standards: 76.

Paz, J. A. M., M. Y. M. Gómez and S. C. Rosas (2017). Análisis sistemático de información de la Norma ISO 25010 como base para la implementación en un laboratorio de Testing de software en la Universidad Cooperativa de Colombia Sede Popayán. Memorias de Congresos UTP.

Verona-Marcos, S., Y. Pérez-Díaz, L. Torres-Pérez, M. D. Delgado-Dapena and C. Yáñez-Márquez (2016). "Pruebas de rendimiento a componentes de software utilizando programación orientada a aspectos." SciELO: 278-285.

Yuan, R., C. H. Salgado, M. Peralta and A. Sánchez (2020). Evaluación de un modelo ontológico basado en

la adecuación funcional de la norma ISO 25010 para la elicitación de requisitos de software. XXVI Congreso Argentino de Ciencias de la Computación (CACIC)(Modalidad virtual, 5 al 9 de octubre de 2020).

Conflict of Interest

The authors of this article authorize the distribution and use of their article.

Authors' contributions

1. Conceptualization: Maidelyn Piñero González
2. Data curation: Aymara Marin Díaz
3. Formal analysis: Leosmay Carrión Estradet
4. Acquisition of funds: -
5. Research: Leosmay Carrión Estradet
6. Methodology: Leosmay Carrión Estradet
7. Project Administration: Aymara Marin Díaz
8. Resources: Amanda Delgado Martínez
9. Software: Leosmay Carrión Estradet y Amanda Delgado Martínez
10. Supervision: Aymara Marin Díaz
11. Validation: Maidelyn Piñero González
12. Visualization: Leosmay Carrión Estradet y Amanda Delgado Martínez
13. Editing - original draft: Leosmay Carrión Estradet y Maidelyn Piñero González
14. Writing - proofreading and editing: Maidelyn Piñero González

Funding

No funding was necessary for the development of the research.