Neutrosophic Optimization and Intelligent Systems

Journal Homepage: sciencesforce.com/nois

Neutrosophic Opt. Int. Syst. Vol. 6 (2025) 6–14

Paper Type: Original Article

Solutions Framework Software Evaluations using Neutrosophic Figures

Hans Daniel Zambrano Campi [1,*], Angel Braulio Martinez Vasquez [2], Vladimir Vega Falcon [3] and Silvia Patricia Chiriboga Velasco [4]

[1] Teaching, University Regional Autonomous of The Andes, Ecuador; ub.hanszambrano@uniandes.edu.ec.
[2] Analyst, Association Latin American of Sciences Neutrosophic, Ecuador; vasmarti10@gmail.com.
[3] Analyst, Investigation Uniandes Ambato, Ecuador; ua.vladimirvega@uniandes.edu.ec.
[4] Teacher, University Regional Autonomous of The Andes, Headquarters Babahoyo, Ecuador; b.silviapcv.caula@uniandes.edu.ec.

Received: 05 Aug 2024 **Revised:** 15 Jan 2025 **Accepted:** 16 Apr 2025 **Published:** 18 Apr 2025

Abstract

This article addresses a key challenge in software development: how to identify opportunities to optimize testing within the Microsoft Solutions Framework (MSF), a widely used framework for managing technology projects. The central question guiding this research is how to detect and prioritize these opportunities in an environment characterized by uncertainty and the subjectivity of human decision-making. To this end, the study introduces the use of neutrosophic numbers, a mathematical tool designed to manage ambiguity and the diverse opinions of experts. This approach seeks to overcome the limitations of conventional methods, which often ignore the uncertainty present in the evaluation of complex processes such as software testing. The relevance of this topic lies in the growing dependence of companies and organizations on robust technological solutions, where software testing plays a crucial role in ensuring quality and functionality. However, previous approaches have lacked adequate tools to integrate uncertainty into their analyses, leaving a gap that this work aims to fill. Applying a methodology based on neutrosophic numbers, the study collects and processes expert judgments, revealing findings that highlight critical areas for improvement in MSF, such as test planning and execution. These results not only enrich the theory by offering an innovative model for decision-making, but also provide practical guidelines that can be directly implemented in real-world projects, thus strengthening the effectiveness and adaptability of software development processes in dynamic contexts.

Keywords: Methodology; Opportunities; Software Testing; Microsoft Solutions Framework; Neutrosophic Numbers; Uncertainty; Software Development.

1 | Introduction

Agile methodologies are a well-established topic in software engineering, generating great enthusiasm and adoption in short-term initiatives with constantly evolving demands. In contrast, the classic approach requires strict delineation of roles, tasks, and products, along with thorough design and record-keeping [1]. The Agile Alliance, a non-profit entity, is dedicated to promoting the principles of agile software development and supporting companies in implementing these ideas. It all started with the Agile Manifesto, a text that encapsulates the essence of this philosophy [2, 3]. According to this manifesto, people and team dynamics are prioritized over procedures and tools, highlighting that human talent is the key element for the success of any technological project. Building a solid team is more valuable than preparing the perfect scenario. Frequently,

Corresponding Author: ub.hanszambrano@uniandes.edu.ec
https://doi.org/10.61356/j.nois.2025.6536

people fall into the error of structuring the environment first and assuming that the members will adjust themselves when the ideal is to consolidate the team and allow it to shape its workspace according to its requirements [4, 5]. The development of functional software prevails over the development of extensive documentation; the guideline is to avoid generating reports unless they are immediately essential for crucial decisions, keeping them brief and focused on the essentials. Cooperation with the client is prioritized over contractual negotiations, promoting a continuous exchange between the client and the developers, which will guide the course of the project and guarantee its success. Likewise, the ability to adapt to changes—whether in requirements, technology, or the team itself—is favored over rigid adherence to a predefined framework [6, 7]. This flexibility in the face of unforeseen events largely determines whether the project will succeed or not, so planning must be adaptable and responsive. These values underpin the twelve principles of the manifesto, traits that distinguish an agile process from a conventional one. The initial two are broad and reflect the core of the agile spirit, while the remaining ones address the flow to follow and the organization of the team, establishing clear objectives and guidelines: satisfying the customer with fast and valuable deliveries, embracing changes for competitive benefit, delivering working software frequently, encouraging daily collaboration between business and developers, driving projects with motivated and supported people, prioritizing direct communication, measuring progress through functional software, maintaining a sustainable pace, ensuring technical excellence, focusing on simplicity, trusting in the autonomy of teams and reflecting periodically to improve effectiveness.

Agile and traditional methodologies, also known as "heavy" methods, present notable differences in their approach and application. While the former is based on practical rules derived from experience in code creation and easily adapts to modifications throughout the project, the latter relies on regulations established by development environment standards and tends to display certain inflexibility in the face of change. Decisions in agile approaches usually arise from within the team, with a more relaxed process guided by basic principles, in contrast to traditional methodologies, which are imposed from outside and governed by a strict framework full of detailed policies. In the contractual area, agile methods avoid rigid agreements or make them more flexible, while traditional methods operate under predefined contracts. Furthermore, agile teams are compact, with fewer than ten members collaborating in a single location, and they generate few document products or roles, prioritizing simplicity. On the other hand, groups in heavyweight methodologies tend to be larger, often geographically distributed, produce a larger number of artifacts, involve multiple roles, and place considerable weight on software architecture, represented by specific models.

2 | Preliminaries

The section presents the basic theory for the comprehension of the proposal of investigation. Describe Microsoft Solutions Framework for the agile application development. It introduces the associated theory of software testers and validation testing. Finally, it describes neutrosophic numbers in the context of the present research for the selection of the methodology.

2.1 | Microsoft Solutions Framework

Microsoft Solutions Framework (MSF) for the Development of Applications Agile, is the proposal of Microsoft for application development using an agile methodology; which incorporates practices to handle quality of service (QoS) requirements, such as performance and security [8, 9].

The phases of the MSF are the following:

- Phase 1 - Strategy and scope
- Phase 2 - Planning and Proof of Concept
- Phase 3 - Stabilization
- Phase 4 – Deployment of the roles in MSF Agile

In the model of the team of MSF, there are no hierarchies, all are equal or important. Roles:

- Analyst of business
- Boss of the project
- Architect
- Developer
- Staff of evidence
- Staff of deployment
- Users experienced

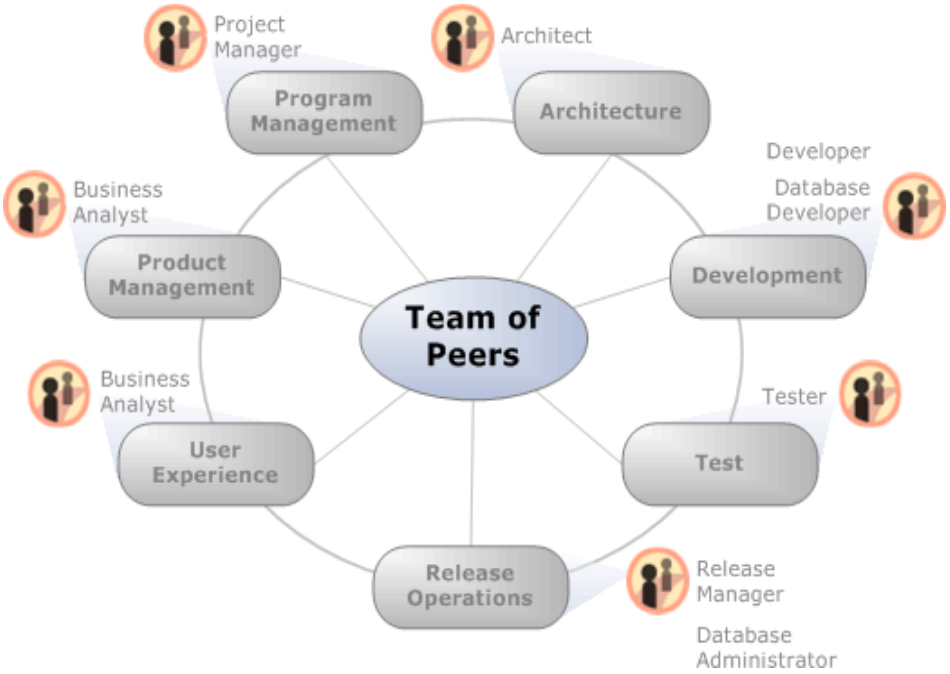

Figure 1. Roles in MSF for the Development of Applications Agile.

This equipment of specialists, with roles different, brings together in representation of all the members involved with the production, use, and maintenance of the product. Each team member, or each role, is responsible for representing the needs specific of all the members of his group and none is more important than the other. These meetings provide the balances and checks necessary to ensure a genuine solution is achieved. Figure 1 shows the composition of the team of specialists.

2.2 | Fitting Rooms of Software

He staff of evidence orders of the area of evidence in the MSF Team Model. His aim major is to find and communicate the issues of the product that could affect negatively his worth. The staff of evidence must understand the context of the project and help other people to base their decisions in said context. An aim clue of the staff of evidence is locating the errors significant that present the product during the phase testing and reporting accordingly. Once a bug is found, it is also up to testing staff to accurately communicate its consequences and describe workarounds that would reduce its impact. They should write easy-to-understand descriptions of the bugs and the steps required to reproduce them. To follow. Also, participate with the rest of the team in the definition of the standards of quality for the product. The purpose of testing is to verify that known functions are performing their functions correctly and to discover new problems.

His flow work of the staff of tests is the following:

- Analysis
- Close errors
- Development of the documentation

- Establishment of the environment test
- Establishment of the process of the project
- Launch of the product
- Proof of a requirement of the customer
- Testing of a requirement of the product

2.3 | Fitting Rooms of Scenarios

A scenario is divided into testing tasks and development tasks and testers are responsible for developing and Execute test cases. Assigning a scenario to be tested indicates that the functionality has been integrated into a framework and is ready to be tested [10, 11]. Validating that a framework reflects the intended functionality of the scenario requires an understanding of the scenario and its boundary conditions; therefore, validation tests should be written to cover the complete functionality as well as the boundary conditions of the scenario and will be executed as bugs are reported.

What to do for test scenarios?

MSF raises a set of activities that should be carried out for prove scenarios:

Define approximation of proof

An approach of proof is a strategy that guides the plan of proof and its execution, to the time that determines the models of quality for the package of the product [12, 13]. The activities of the approach of proof are a point of start for the plan of proof early in the project, but it evolves and changes with this. An approximation test should include a mixture of techniques, including the manual and the evidence automated, and before each iteration, the document of approach of proof has to be put on to the day to reflect the goals of the com - checking the iteration and test data that will be used.

The sub-activities defined for this activity are:

- Determine the context of the project: identify the risks of the project and the users that these could affect, so as the situations specials that could impact the level of testing necessary. It must be determined what is at risk and its impact, in case the product fails.
- Determine the test mission: Identify the project goals to be met through testing, and consulting with the architect and business analyst on technical uncertainties and user risks.
- Evaluate potential testing techniques: The available testing tools and skills of the testing team should be evaluated to determine the testing techniques that are possible and appropriate for the project.
- Define test metrics: Use the project context, test mission, and testing techniques to determine test metrics. These metrics will include thresholds for various types of tests (load, performance, etc.) or the percentage of automated tests.

2.4 | Tests Validation

The evidence of validation assures the functionality of the system, they take a view of the "box “black” of the application and focus on the most important areas for the end user to check the functionality corresponding with it written in the scenery [14, 15]. Writing the cases of evidence for the evidence of Validation helps testers identify problems using testing mechanisms that mimic the real world.

For writing a proof of validation has to take into account the following aspects:

- Identify the area and atmosphere of the test: Isolate the area where will run the test. The evidence of iteration is the set of cases of proof automated that they run after the evidence validation of functionality. However, a feature does not have to pass this test to be successful. Tests can be run as part of iteration testing if they are automated.

- Identify the details of the flow of the cases of proof: Identify the data of proof required for each case of test, learning in the report of the approach of test, like, the restrictions and conditions of limit for the cases of evidence required in the tasks of proof. Check yeah the cases of Tests can be automated and identify procedural steps for the scenario flow.
- Write cases of proof: Write the documentation of the evidence for the manuals of cases of test and automated test cases for iteration testing.
- Other activities to execute are: the selection of a case of proof for running it, discovering possible bugs, and carrying out evidence exploratory refers in the spot dedicated to the evidence of the requirements of quality of service.

2.5 | Numbers Neutrosophics for the Determination of Opportunities in the Application of Evidence

The determination of opportunities in the application of software testing can be modeled as a multi-criteria decision-making problem [16, 17]. From which a set of alternatives are possessed that represent the evidence of the development of software $= \{A\,1, \ldots A\,n\}, n \geq 2$; which them they perform an assessment based on the set of criteria $C = \{C\,1, \ldots C\,m\}, m \geq 2$ that characterize the project.

The solution is defined with a spectrum that represents the true preference of using a type of evidence with a degree of falsity and a degree of denial. Problems of this nature have been modeled as a neutrosophic problem.

Neutrosophic allows the representation of neutrality, it was proposed by Smarandache [18]. It represents the basis for a series of mathematical theories that generalize classical theories and diffuses such as the con- sets Neutrosophics and the logic neutrosophic. A number neutrosophic (N) represents of the following shape [19]: $N = \{(, I, (n) : T, I, n \subseteq [0,1]\}$, a valuation Neutrosophic is a mapping of a group of proportional formulas too, that is, for each sentence p we have:

$$v\,(p) = (T, I, F) \quad (1)$$

Where:

T: represents the dimension of the space that represents the true,
I: represents the falsehood,
F: represents the indeterminacy.
A valuation neutrosophic is a mapping of a cluster of formulas proportional to N, whereby each sentence p we have:

$$v\,(p) = (T, I, F)$$

3 | Implementation of Evidence about the Quality of the Service

To validate that a structure reflects the planned restrictions in the requirements of quality of service, it needs knowledge further there of the restriction and that the evidence of performance, safety, effort, and loads are completed and none are blocked. MSF proposes a set of activities that must be performed to test the quality of service requirements: The approach to the test was already described when we discussed scenario testing, in which it was said that it is the step that precedes the creation of the plan of tests. The plan Testing allows you to specify what you want to test and how to run and measure the progress of those tests [20, 21].

To define a proof of performance there is perform the following Sub-activities: Understand the objective of the test.

Specify the configuration of the proof: The variables of configuration include the hardware, system operating, software, and other features whose use is important for running tests. Each test configuration can represent an entry in the test matrix.

Design the proof: must map the conditions of the test including the requirements previously and the programmed scenario that must be reviewed to determine in which areas performance may be critical.

The evidence of security or evidence of penetration employs the threats found in the process of the threat model to simulate an attempt by the adversary to attack the product. This form of testing can be divided into three parts: exploration, identification of the defect, and exploitation. The evidence of penetration can discover new vulnerabilities that convert into requirements of security or errors in the tried of block entry points and subsequent access to assets.

This shape of proof requires skills special in thinking and acting as the adversary. The Stress tests determine the breaking points of an application and put the application beyond its upper limit in that the resources are saturated. They are used to identify the boundaries superiors of the burden of the application where the application response has degraded to an unacceptable level or has failed.

A stress test is a type of performance test. They can be used to predict application behavior and validate the stability of application and reliability through execution from load testing over an extended period.

A load test is another type of performance test. Load tests help ensure that the application fulfills its requirements of quality of service and low conditions of burden. To execute evidence of load, it is common to focus on high-traffic areas, the 20% of the application that is used 80% of the time.

Testing is a systematic way of testing a product, the objective is to discover new scenarios and new requirements for quality of service. Is important to define a range of time limits for the test and keep a log.

Two other activities to be performed by the testers are: Selecting and running a test case and discovering a bug.

4 | Main Results

This section describes an example to demonstrate the applicability of the proposed method in the case of the selection of the type of test. The example presents the fundamental elements summarized to facilitate readers' understanding. The main evaluation criteria taken into account for the test selection consisted of the following five indicators:

C_1 Redundancy,
C_2 Complexity,
C_3 Dynamism,
C_4 Specialization,
C_5 Staff.
Determination of the weights of the criteria evaluative.

From the consultation carried out with experts were obtained the vectors of importance W attributed to each indicator. Table 1 shows the values resulting from the activity.

Table 1. Weights are certain for the indicators.

Indicators	Weights W
1	0.63
2	0.85
3	0.74
4	0.88
5	0.94

They perform a prosecution of the evaluations about the compliance of the criteria.

To leave off the evaluations expressed by the experts about the behavior of the indicators in the case study, the averaged preferences are obtained by indicators as expressed in Table 2.

Table 2. Result of the preferences.

Criteria	C1	C2	C3	C4	C5
Assessment	MD	M	MA	B	B

To leave off the result of the preferences he obtained a vector preferably such as is expressed:

$$S = [0.80, 0.7, 0.84, 0.75, 0.66]$$

Finally, for his case of study, he obtained an assessment general: E = 0.75

The result expresses that the recommendation produces the utilization of dynamic tests.

To leave the selection of the proof dynamic, starts his process of proof.

For the specific case of the evidence can say that MSF Agile grants great importance at tests and allows an exploitation of the tools integrated into the Visual Studio (VS), although they can be employed by other tools.

The evidence they can run inside and out of the VS. Net. Internally: Test Manager and Test Results Externally: MS Build (script)

VS Team Suite allows the creation of Work Items (tasks I bug) associated with the execution of the tests.

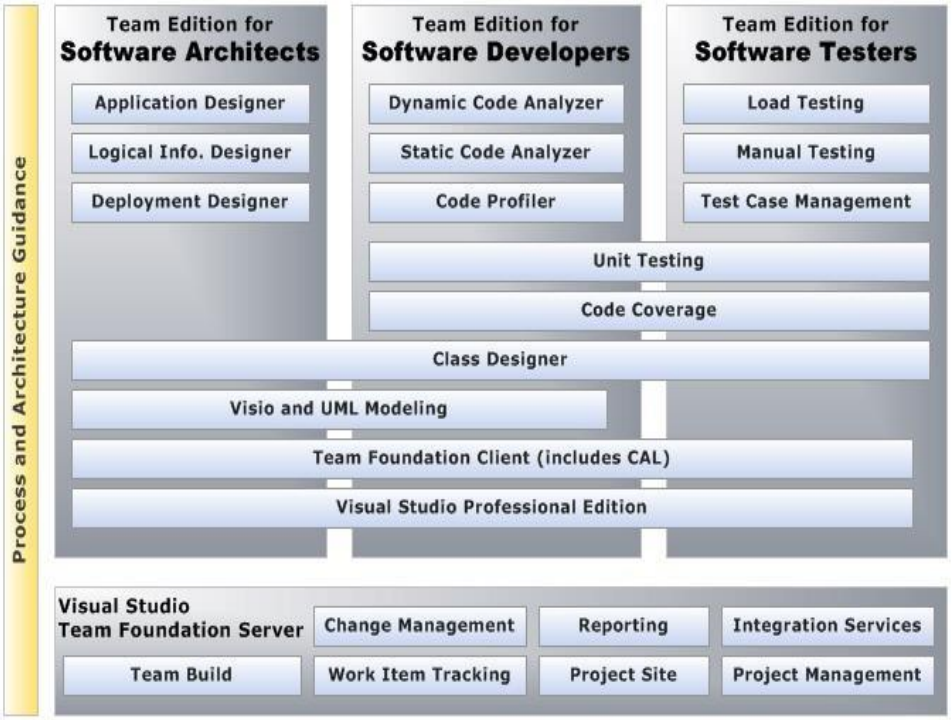

Figure 2. Visual Studio Team Suite as a tool of medium.

VSTS has functionalities for the realization of evidence of applications Web (Figure 2). From Visual Studio, You can record navigation and then add rules to validate the responses. It also offers load test generation capabilities by defining scenarios.

5 | Conclusion

The determination of opportunities in the application of tests of software represents a task important to the initial process of development. This knowledge constitutes a problem of decision that can be modeled using neutrosophic numbers.

Microsoft Solutions Framework represents an methodology agile that provides a linkage further with customers and looks for all products are deliverables. It is a flexible methodology, easy to handle, and tends to simplify project management for small, short-term applications.

Acknowledgments

The author is grateful to the editorial and reviewers, as well as the correspondent author, who offered assistance in the form of advice, assessment, and checking during the study period.

Author Contribution

All authors contributed equally to this work.

Funding

This research has no funding source.

Data Availability

The datasets generated during and/or analyzed during the current study are not publicly available due to the privacy-preserving nature of the data but are available from the corresponding author upon reasonable request.

Conflicts of Interest

The authors declare that there is no conflict of interest in the research.

Ethical Approval

This article does not contain any studies with human participants or animals performed by any of the authors.

References

[1] TO. N. Cadavid, J. D. F. Martinez, and J. M. Vélez, "Revision of methodologies agile for he development of software," *Prospective,* vol. 11, No. 2, pp. 30-39, 2013.

[2] J. H. Canós , and M. C. P. P. Letelier, "Methodologies agile in he development of software," 2012.

[3] Smarandache, F., & Leyva- Vazquez, MY (2024). Preface: Special Issue: Proceedings of the International Conference Advances in SuperHyperStructures and Applied Neutrosophic Theories, Universidad de Guayaquil, Ecuador, 28-29 November 2024. *Neutrosophic Sets and Systems , 74* . https://doi.org/10.5281/zenodo.13989116

[4] Barzola-Monteses, J., Caicedo-Quiroz, R., Parrales-Bravo, F., Medina-Suarez, C., Yanez-Pazmino , W., Zabala-Blanco, D., & Leyva- Vazquez , MY (2024). A Neutrosophic Framework for Artificial Immune Systems. *Neutrosophic Sets and Systems, 74* , 210-226. https://doi.org/10.5281/zenodo.14418335

[5] Cevallos-Torres, L., Martínez, R., Caicedo-Quiroz, R., Hernández-Magallanes, R., Iturburu -Salvador, D., Parrales-Bravo, F., & Leyva-Vázquez, M. (2024). Assessment of Academic Integrity in University Students Using a Hybrid Fuzzy-Neutrosophic Model under Uncertainty. *Neutrosophic Sets and Systems, 74* , 267-274. https://doi.org/10.5281/zenodo.14418553

[6] Cevallos-Torres, L., Núñez- Gaibor , J., Leyva- Vasquez , M., Gómez-Rodríguez, V., Parrales-Bravo, F., & Hechavarría-Hernández, J. (2024). NCC: Neutrosophic Control Charts, a didactic way to detect Cardiac Arrhythmias from reading Electrocardiograms. *Neutrosophic Sets and Systems, 74* , 441–456. https://doi.org/10.5281/zenodo.14420234

[7] P. Letelier, and MC Penadés , "Agile methodologies for software development: eXtreme Programming (XP)," 2006.

[8] EITHER. T. Gómez, PPR López, and JS Bacalla , "Criteria of selection of methodologies of development of software,"*Industrial data,* vol. 13, No. 2, pp. 70-74, 2010.

[9] TO. EITHER. Duarte, and M. Rojas, "The methodologies of development agile as a chance for the engineering of educational software," *Advances in Systems and Informatics Magazine,* vol. 5, no. 2, pp. 159-171, 2008.

[10] M. Figueroa, " MeISE : Methodology of engineering of software educational," Magazine International International International Journal of Engineering Education Engineering Education ISSN, vol. 1116, 1940.

[11] Smarandache, F. (2022). Extension from Soft Set to Hypersoft Set, and then to Plithogenic Hypersoft Set. *Neutrosophic Computing and Machine Learning, 25* , 103-106. https://doi.org/10.5281/zenodo.7519268

[12] Smarandache, F. (2022). Introduction to Super - Hyper -Algebra and Neutrosophic Super - Hyper -Algebra. *Neutrosophic Computing and Machine Learning, 20* , 1-6. https://doi.org/10.5281/zenodo.6612313
[13] Smarandache, F. (2021). Introduction to Plitogenic Logic. *Neutrosophic Computing and Machine Learning, 18*, 1-6. https://doi.org/10.5281/zenodo.5525533
[14] M. Arias, TO. Lopez, and J. Honmy , "Methodology dynamics for he development of software educational," 2015.
[15] S. Machiraju, and R. Modi , "Developing Bots with Microsoft Bots Framework," 2018.
[16] G. EITHER. Tashchiyan , AV Sushko , and S. V. Grichin , "Microsoft Business Solutions-Axapta ace to basis for automated monitoring of high technology products competitiveness." p. 012065.
[17] C. Tascon , and H. Domínguez, "Analysis to the utility of the technique of scenarios in the elicitation of requirements,"*Magazine Antioquian of the Sciences Computational,* vol. 7, No. 1, 2017.
[18] JV Jimeno Flores, C. Hernández, and G. Milckar , "Methodology to measure the performance level of software testers at G & V Servigen SAC," 2018.
[19] E. Serna, R. Martínez, P. Tamayo, and IU de Envigado, "A review of the reality of software testing automation," *Computing and Systems,* vol. 23, no. 1, pp. 169–183, 2019.
[20] CC Villada Zapata, CA Cifuentes Hincapie , VM Delgado Pena, and OH Rojas García, "Deepening website software testing" MercadoLibre ," 2019.
[21] O. Mar, and B. Bron , "Base Counselor of the Action for the development of internships in a System of Distance Laboratories " *Scientific Journal,* vol. 2, no. 29, pp. 140-148, 2017.