Neutrosophic Sets and Systems

Volume 37 Article 32

1-1-2020

# Opportunities for Software Testing using Neutrosophic Numbers

Alberto Santillán Molina

Wilson Alfredo Cacpata Calle

Javier Dario Bosquez Remache

Neutrosophic Sets and Systems
NSS {Special Issue: Impact of Neutrosophy in solving the Latin American's social problems}, Vol. 37, 2020
University of New Mexico

# The application of Microsoft Solution Framework Software Testing using Neutrosophic Numbers

**Alberto Santillán Molina[1], Wilson Alfredo Cacpata Calle [2], Javier Dario Bosquez Remache [3]**

[1] Docente de la carrera de Derecho, Universidad Regional Autónoma de los Andes (UNIANDES), Avenida La Lorena, CP 230150, Santo Domingo de los Tsáchilas, Ecuador, Ecuador. E-mail: us.albertosantillan@uniandes.edu.ec
[2] Docente de la carrera de Derecho, Universidad Regional Autónoma de los Andes (UNIANDES), Avenida La Lorena, CP 230150, Santo Domingo de los Tsáchilas, Ecuador, Ecuador. E-mail: us.wilsoncacpata@uniandes.edu.ec
[3] Docente de la carrera de Derecho, Universidad Regional Autónoma de los Andes (UNIANDES), Avenida La Lorena, CP 230150, Santo Domingo de los Tsáchilas, Ecuador, Ecuador. E-mail: us.javierbosquez@uniandes.edu.ec

**Abstract.** For software development companies, deciding which development methodology to use when starting a project is in many cases an important task to perform. The selection of development methodology can be modeled as a decision-making problem. Neutrosophy is a philosophical current that starts from Paradoxism. In this article, we describe a methodology selection proposal using neutrosophic numbers, from which the development methodology proposal is decided. The method allows taking into account the indeterminacy in decision-making, in addition to the use of linguistic terms that are more appropriate than numerical ones. The applicability of the proposal is confirmed through a demonstrative example. The paper ends with the conclusions and recommendations for future work.

**Keywords:** Agile Methodologies, Microsoft Solutions Framework, validation tests, neutrosophic numbers.

## 1. Introduction

Agile methodologies are an established topic in software engineering that have aroused interest and acceptance in their application in short-term projects with changing requirements, the traditional scheme requires a rigorous definition of roles, activities and artifacts accompanied by modeling and detailed documentation [1]. *The Agile Alliance* is a non-profit organization dedicated to promoting the concepts related to agile software development and helping organizations to adopt these concepts. The starting point was the Agile Manifesto, a document that summarizes the agile philosophy[2],[3].

According to the Manifesto, the individual and the interactions of the development team on the process and tools are valued. People are the main success factor of a software project. It is more important to build a good team than to build the environment. Many times people make the mistake of building the environment first and expecting the team to adapt automatically. It is better to create the team and have it configure its own development environment based on your needs [4, 5].

Developing software that works, rather than getting good documentation. The rule of thumb is not to produce documents unless they are immediately needed to make an important decision. These documents should be short and focus on the fundamentals. Collaboration with the client is more than negotiating a contract. It is proposed that there is a constant interaction between the client and the development team. This collaboration between both the client and the team will be what guides the progress of the project and ensures its success.

It is about reacting to changes, rather than strictly following a plan [6, 7]. The ability to react to changes that may arise throughout the project (changes in requirements, technology, equipment, etc.) also determines the success or failure of the project. Therefore, planning should not be strict but flexible and open.

The above values inspire the twelve principles of the manifesto. These are characteristics that differentiate an agile process from a traditional one. The first two principles are general and summarize much of the agile spirit. The rest have to do with the process to be followed and the development team, in terms of goals to follow and its organization. The principles are:

The priority is to satisfy the customer through early and continuous releases of software that adds value.

- Welcome changes. Changes are captured so that the customer has a competitive advantage.
- Frequently deliver software that runs from two weeks to two months, with the shortest possible time interval between deliveries.
- The business team and the developers must work together throughout the project.
- Build the project around motivated individuals. Give them the environment and support they need and trust them to get the job done.
- Face-to-face dialogue is the most efficient and effective method of communicating information within a development team.
- Software that works is the main measure of progress.
- Agile processes promote sustainable development. Promoters, developers and users should be able to maintain constant peace.
- Permanent attention to technical quality and good design improves agility.
- Simplicity is essential.
- The best architectures, requirements, and designs come from self-organizing teams.
- At regular intervals, the team reflects about how to become more effective, and adjusts its behavior accordingly.

| Agile methodology | Traditional methodology |
|---|---|
| Based on heuristics from code production practices | Based on standards followed by the development environment. |
| Specially prepared for changes during the project | Some resistance to change |
| Internally imposed by the team | Externally imposed |
| Less controlled process, with few principles | Much more controlled process, with numerous policies and regulations |
| There is no traditional contract or at least quite flexible | There is a predetermined contract |
| Small groups (-10 members) and working in the same place | Large groups and possibly distributed |
| Few artifacts | More artifacts |
| Few roles | More roles |
| Less emphasis on software architecture | Software architecture is essential and is expressed through models |

**Table 1.** Comparison between agile and "heavy" methodologies

The present research proposes a method based on neutrosophic numbers [8-10]such that the different alternatives based on scenarios are evaluated by a group of experts according to evaluation criteria. The method includes the indeterminacy of decision-making, in addition to allowing calculation with the help of a linguistic scale, which is more suitable for human evaluators.

The article consists of the following sections: Section 2 delves into different aspects of the Microsoft Solutions Framework, recalls the concepts of the neutrosophic number and introduces the method that we will apply. Section 3 contains some additional details and the application of the method in an example. At the end of the paper, we state the conclusions.

## 2 Preliminaries

The section presents the theoretical bases for understanding the research proposal. Describes the Microsoft Solutions Framework for Agile Application Development. Introduces the associated theory about software testers, validation testing. Finally, we describe the neutrosophic numbers in the context of the present investigation for the selection of the methodology [11-13].

### 2.1 Microsoft Solutions Framework

Microsoft Solutions Framework (MSF) for the Development of Agile Applications, is Microsoft's proposal for the development of applications using an agile methodology; which incorporates practices to manipulate quality of service (QoS) requirements, such as performance and security[14, 15].

The phases of the MSF are:

Phase 1 - Strategy and scope

Phase 2 - Planning and proof of concept

Phase 3 - Stabilization

Phase 4 - Deployment

The roles in MSF Agile. In the MSF team model there are no hierarchies, they are all equally important.

The recommended roles are:

- Business analyst

- Project manager
- Architect
- Developer
- Testing staff
- Deployment personnel
- Experienced users

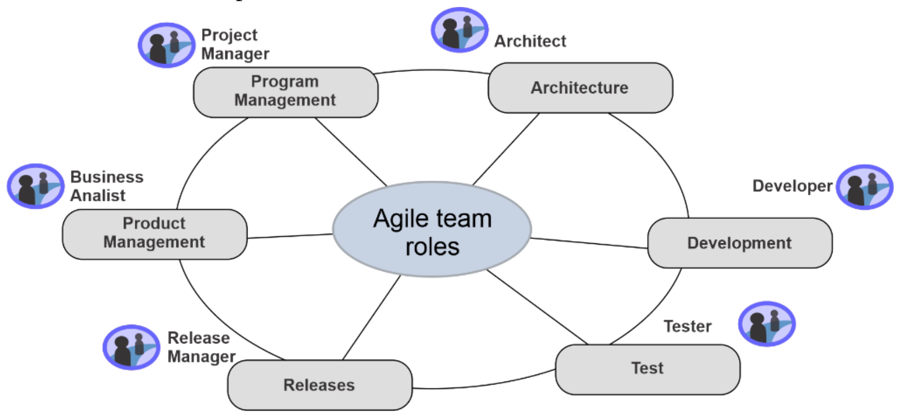

**Figure 1.** MSF Roles for Agile Application Development

This team of specialists, with different roles, meets on behalf of all the members involved with the production, use and maintenance of the product. Each member of the team, or each role, is responsible for representing the specific needs of all members of their group and none is more important than another. These meetings provide the necessary balances and checks to ensure that a true solution is produced. Figure 1 shows the composition of the team of specialists.

## 2.2 Software Testers

The test staff manages the test area in MSF Team Model. Their primary goal is to find and report product issues that could adversely affect its value. Test personnel must understand the context of the project and help others base their decisions on that context. A key objective of the test personnel is to locate the significant errors that the product may present during the test phase and to report accordingly. Once a bug is found, it is also up to the testing staff to accurately communicate its consequences and describe workarounds that would reduce its impact. They should write descriptions of the errors and the steps necessary to reproduce them, easy to understand and to follow. They also participate with the rest of the team in defining the quality standards for the product. The purpose of the tests is to see if known functions are working correctly and to discover new problems.

The test staff workflow is as follows:

- Analysis
- Close errors
- Documentation development
- Establishment of test environments
- Establishment of the project process
- Product Launch
- Proof of a customer requirement
- Checking a product requirement

## 2.3 Scenario testers

A scenario is divided into test tasks and development tasks and testers are in charge of developing and executing the test cases. Assigning a scenario to test indicates that the functionality has been integrated into a framework and is ready to be tested.[16], [17]. Validating that a structure reflects the intended functionality of the scenario requires an understanding of the scenario and its boundary conditions; so the validation tests should be written to cover the full functionality as well as the boundary condition of the scenario and will be executed while bugs are reported
What to do to test scenarios?
MSF proposes a set of activities that must be carried out to test scenarios:
Define test approach

A test approach is a strategy that guides the test plan and its execution, while determining the quality models for packaging the product.[18], [19]. The test approach activities are a starting point for the project's anticipated test plan, but it evolves and changes with it. A test approach should include a mix of techniques, including manual and automated tests, and prior to each iteration, the test approach document should be updated to reflect the goals of the iteration testing and the test data that will be employees.

The sub-activities defined for this activity are:

- Determine the context of the project: The risks of the project and the users that these could affect are identified, as well as the special situations that could affect the level of verification necessary. What is at risk and its impact, should the product fail, must be determined.
- Determine the mission of the test: The goals of the project to be satisfied through the tests are identified, consulting with the architect and business analyst on technical uncertainties and user risks.
- Evaluate possible testing techniques: The tools available for testing, as well as the skills of the test team, should be evaluated to determine possible and appropriate testing techniques for the project.
- Define test metrics: You should use the project context, test mission, and test techniques to determine test metrics. These metrics will include the thresholds for the various types of tests (load, throughput, etc.) or the percentage of automated tests.

## 2.4 Validation tests

The validation tests ensure the functionality of the system, take a "black box" view of the application and focus on the areas most important to the end user in order to check the functionality corresponds to what is written on the scenario[20, 21]. Writing test cases for validation tests helps testers identify problems using test mechanisms that mimic the real world.

To write a validation test, you must consider the following aspects:

- Identify the test area and environment: Isolate the area where the test will be run. Iteration tests are the set of automated test cases that run after the functionality validation tests. However, a feature does not have to pass this test to be successful. Tests can be run as part of iteration tests if they are automated.
- Identify the details of the test case flow: Identify the test data required for each test case, based on the test approach report, as well as the constraints and boundary conditions for the test cases required in the tasks test. Check if test cases can be automated and identify procedural steps for the scenario flow.
- Write test cases: Write test documentation for test case manuals and automated test cases for iteration tests.
- Other activities to be executed are: the selection of a test case to run it, discovering a possible bug and carrying out exploratory tests refers to the point dedicated to testing the quality of service requirements.

## 2.5 Neutrosophic numbers for the determination of opportunities in the application of tests

The determination of opportunities in the application of software tests can be modeled as a multi-criteria decision-making problem[22, 23]. From which there is a set of alternatives that represent software development tests $\mathrm{A} = \{\mathrm{A}_1, \dots \mathrm{A}_\mathrm{n}\}$, $\mathrm{n} \geq 2$; which are evaluated based on the set of criteria $\mathrm{C} = \{\mathrm{C}_1, \dots \mathrm{C}_\mathrm{m}\}$, $\mathrm{m} \geq 2$ that characterize the project [10, 24, 25].

The solution is defined with a spectrum that represents the true preference to use a test type with a degree of falsehood and a degree of denial. Problems of this nature have been modeled as a neutrosophic problem.

Neutrosophy is a philosophical current that starts from Paradoxism [26]. In the 1980s, the international movement called Paradoxism developed [27], based on contradictions in science art, was founded by Romanian author Florentin Smarandache currently living in the United States who later extended it to Neutrosophy, based on contradictions and their neutrals with multiple practical applications [26, 28-31].

Neutrosophy allows the representation of neutrality, it was proposed by Smarandache[32]. It represents the basis for a series of mathematical theories that generalize classical and fuzzy theories such as neutrosophic sets and neutrosophic logic. A neutrosophic number (N) is represented as follows [11-13, 33, 34]:

Let $N = \{(T, I, F): T, I, F \subseteq [0, 1]\}$, a neutrosophic valuation is a mapping of a group of propositional formulas over $N$, that is, for each proposition p we have

$$v\,(p) = (T, I, F) \tag{1}$$

Where:

T: represents the degree of truth,

I: represents the degree of indeterminacy,

F: represents the degree of falsehood.

The main evaluation criteria that are taken into account for the selection of the test are made up of the following 5 indicators:

$\mathrm{c}_1$Redundancy,

$\mathrm{c}_2$Complexity,

$\mathrm{c}_3$Dynamism,

$\mathrm{c}_4$Specialization,

$c_5$Personal.

The linguistic terms that are used to measure the importance of such criteria are summarized in Table 2.

| Linguistic term | Value |
|---|---|
| Not important | (0.10,0.90,0.90) |
| Less important | (0.20,0.85,0.80) |
| Slightly important | (0.30,0.75,0.70) |
| Somewhat important | (0.40,0.65,0.60) |
| Average importance | (0.50,0.50,0.50) |
| Important | (0.60,0.35,0.40) |
| Very important | (0.70,0.25,0.30) |
| Strongly important | (0.8,0,15,0.20) |
| Very strongly important | (0.9, 0.1, 0.1) |
| Extremely important | (1,0,0) |

**Table 2.** Domain of values to assign weight to the criteria.

The linguistic terms [35, 36] we used to evaluate the criteria are summarized in Table 3..

| Linguistic term | Neutrosophic number |
|---|---|
| Extremely good (EG) | (1,0,0) |
| Very Very good (VVG) | (0.9, 0.1, 0.1) |
| Very good (VG) | (0.8,0.15,0.20) |
| Good (G) | (0.70,0.25,0.30) |
| Medium good (MDG) | (0.60,0.35,0.40) |
| Medium (M) | (0.50,0.50,0.50) |
| Medium bad (MDB) | (0.40,0.65,0.60) |
| Bad (B) | (0.30,0.75,0.70) |
| Very bad (VB) | (0.20,0.85,0.80) |
| Very Very bad (VVB) | (0.10,0.90,0.90) |
| Extremely bad (EB) | (0,1,1) |

**Table 3.** Linguistic terms used.

1. If it is the group of experts who evaluate the software, then we have the following steps to carry out the evaluation of the alternatives: $E = \{E_1, E_2, \cdots, E_l\}$
2. The experts evaluate the importance of each criterion according to the scale that appears in Table 2. Let $\widetilde{W_i} = (\widetilde{w}_{i1}, \widetilde{w}_{i2}, \cdots, \widetilde{w}_{im})\widetilde{w}_{ij}$ (i = 1,2, .., l) be the vector of the evaluations provided by the i-th expert on the criteria, where $\widetilde{w}_{ij}$ is the element of Table 2 such that the i-th expert selected the degree of importance of the j-th criterion.

The aggregation vector of the weights is calculated by the following formula:

$$\widetilde{W} = \left(\frac{\sum_{i=1}^{l}\widetilde{w}_{i1}}{l}, \frac{\sum_{i=1}^{l}\widetilde{w}_{i2}}{l}, \cdots, \frac{\sum_{i=1}^{l}\widetilde{w}_{im}}{l}\right) \quad (2)$$

3. The experts evaluate the alternatives regarding to the criteria according to the linguistic scale that appears in Table 3. Let us call $\tilde{x}_{ijk}$ the element of Table 3, which corresponds to the evaluation given by the i-th expert according to the j-th criterion on the alternative k-th.
4. For each alternative, we calculate:

$$\widetilde{b}_k = \frac{\sum_{i=1}^{l}\sum_{j=1}^{m}\widetilde{w}_j \wedge_N \tilde{x}_{ijk}}{lm} \quad (3)$$

Where is the j-th element of the vector, while is the n-norm defined by Equation 4.

$$(T_1, I_1, F_1) \wedge_N (T_2, I_2, F_2) = \left(min(T_1, T_2), max(I_1, I_2), max(F_1, F_2)\right) \quad (4)$$

For each alternative, the score of the neutrosophic number obtained in Equation 3 is calculated, which is a crisp value that is calculated by formula 5.

$$s(\tilde{b}_k) = \frac{1}{3}(2 + T_k - I_k - F_k) \tag{5}$$

Where, .$\tilde{b}_k = (T_k, I_k, F_k)$

The preferred alternatives are those with a higher value of $s(\tilde{b}_k)$

## 3 Implementation of quality of service (QoS) tests

To validate that a structure reflects the constraints provided in the quality of service requirements, knowledge beyond the constraint is needed and that the performance, safety, stress and load tests are completed and none are blocked. MSF proposes a set of activities that must be carried out to test the quality of service requirements: The test approach was already described when we dealt with the scenario tests, in which it was said that it is the step that precedes the creation of the plan of tests. The test plan allows you to specify what you want to test and how to run and measure the progress of those tests.

To define a performance test, the following sub-activities must be carried out:

Understand the purpose of the test

Specify test settings: Configuration variables include hardware, operating system, software, and other features that are important to use in running tests. Each test configuration can represent one test matrix entry. See figure 2.

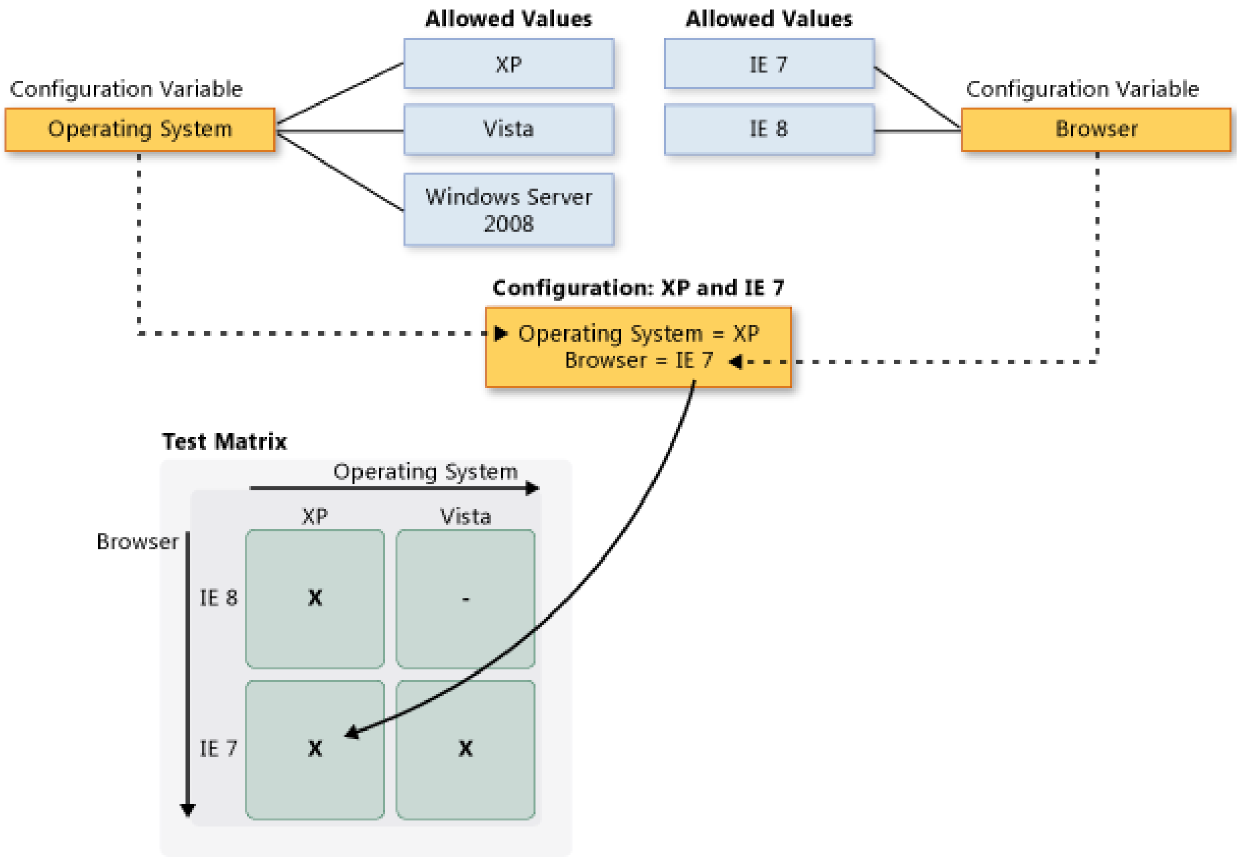

**Figure 2.** Test configuration variables.

Design the test: The test conditions should be mapped including the prerequisites and the scheduled scenario that needs to be reviewed to determine in which areas performance may be critical.

Security testing or penetration testing uses the threats found in the threat modeling process 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. Penetration testing may discover new vulnerabilities that become security requirements or errors in the attempt to block entry points and subsequent access to assets.

This form of testing requires special abilities to be able to think and act like the adversary. Stress testing determines an application's breakpoints and pushes the application beyond its upper limit where resources are saturated. They are used to identify the upper limits of application load where the application response has degraded to an unacceptable level or has completely failed.

A stress test is a type of performance test. They can be used to predict application behavior and to validate an application's stability and reliability by running load tests over an extended period of time.

A load test is another type of performance test. Load testing helps to ensure that the application meets your

QoS requirements under load conditions. When running load tests, it is common to focus on high traffic areas, 20% of the application being used 80% of the time.

Exploratory testing is a systematic way to test a product, the objective is to discover new scenarios or new service quality requirements. It is important to define a time limit range for the test and to keep a log.

Two other activities to be performed by testers are: Select and run a test case and discover a bug.

## Example 1:

This section describes an example to demonstrate the applicability of the proposed method in a test type selection case. The example presents the fundamental elements synthesized to facilitate the understanding of the readers.

After the consultation with 3 experts, the vectors of importance W attributed to each indicator were obtained. Table 4 shows the resulting activity values.

| Indicators | W |
|---|---|
| 1 | (0.63,0.33,0.37) |
| 2 | (0.85,0.12,0.15) |
| 3 | (0.74,0.20,0.26) |
| 4 | (0.88,0.13, 0.12) |
| 5 | (0.94,0.02, 0.06) |

**Table 4.** Weights determined for the indicators.

A processing of the evaluations on the fulfillment of the criteria is carried out.

In this example, we will compare three alternatives that are evaluated according to the 5 criteria by the 3 experts. This can be seen in Table 5, 6 and 7, corresponding to the evaluations by each expert, respectively.

| Criteria | **C1** | **C2** | **C3** | **C4** | **C5** |
|---|---|---|---|---|---|
| A1 | MDG | M | B | G | G |
| A2 | VG | MDG | M | B | VB |
| A3 | MDB | M | M | M | MDG |

**Table 5.** Result of the preferences of expert 1.

| Criteria | **C1** | **C2** | **C3** | **C4** | **C5** |
|---|---|---|---|---|---|
| A1 | M | MDG | VB | VG | VG |
| A2 | G | M | MDG | MDB | B |
| A3 | M | MDG | M | M | M |

**Table 6.** Result of the preferences of expert 2.

| Criteria | **C1** | **C2** | **C3** | **C4** | **C5** |
|---|---|---|---|---|---|
| A1 | G | MDG | MDB | VG | G |
| A2 | G | M | MDG | VB | B |
| A3 | M | MDB | M | MDG | MDG |

**Table7.** Result of the preferences of expert 3.

From the result of the preferences, three neutrosophic numbers were obtained, one for each alternative, these are the following:

$\tilde{b}_1 = (0.58867, 0.38867, 0.41133)$,,.$\tilde{b}_2 = (0.45933, 0.54267, 0.54067)$ $\tilde{b}_3 =$

$(0.51333, 0.48000, 0.48667)$

The scores of each of the vectors by alternatives yielded this result:

$s(\tilde{b}_1) = 0.59622$, $s(\tilde{b}_2) = 0.45867$, and $s(\tilde{b}_3) = 0.51556$. Then A1 is preferred, then A2 and lastly A3.

The result expresses that the recommendation is towards the use of dynamic tests.

From the selection of the dynamic test, the test process starts.

For the specific case of testing, it can be said that MSF Agile grants great importance to testing and allows use of the tools integrated into Visual Studio (VS), although other tools can be used as well.

Tests can be run inside and outside VS.Net.

Internally: Test Manager and Test Results

Externally: MSBuild (script)

VS Team Suite allows the creation of WorkItems (tasks and/or bugs) associated with the execution of the tests

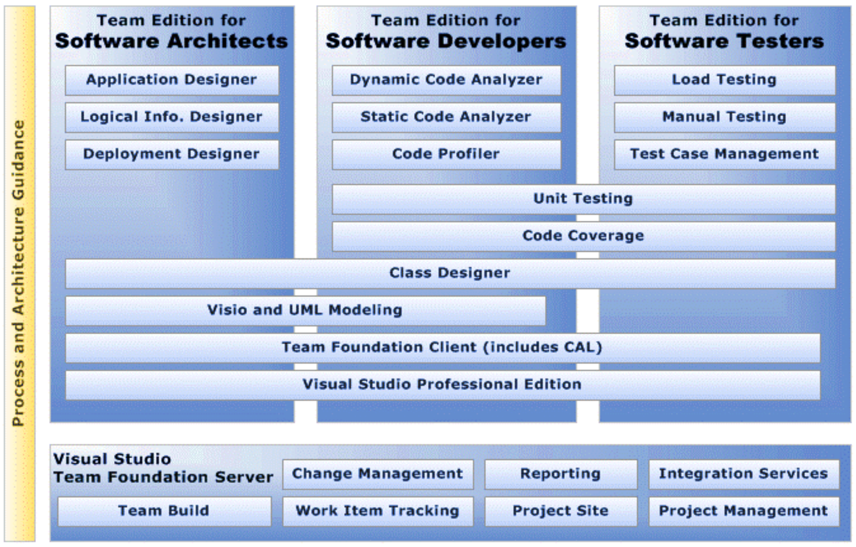

**Figure 3.** Visual Studio Team Suite as a support tool

VSTS has functionalities for testing Web applications. From Visual Studio you can record a navigation to later add rules to validate the responses. It also has capabilities for generating load tests by defining scenarios.

## 4. Conclusions

Determining opportunities in the software testing application represents an important task in the early development process. This knowledge constitutes a discrete decision problem that can be modeled using neutrosophic numbers.

*Microsoft Solutions Framework* represents an agile methodology that provides a closer relationship with customers and seeks that all products are deliverable. A flexible, easy-to-use methodology tends to simplify project management for small and short-term applications. In this article, a multi-criteria decision-making method was proposed for the measurement of alternatives on the Microsoft Solutions Framework methodology. The method allows taking into account indeterminacy in decision-making, in addition to the use of linguistic terms that are more appropriate than numerical terms. As future work, the use of a consensus-based group decision-making model is planned.

## References

1. Cadavid, A.N., J.D.F. Martínez, and J.M. Vélez, *Revisión de metodologías ágiles para el desarrollo de software*. Prospectiva, 2013. **11**(2): p. 30-39.
2. Canós, J.H. and M.C.P.P. Letelier, *Metodologías ágiles en el desarrollo de software*. 2012.
3. Letelier, P. and M.C. Penadés, *Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP)*. 2006.
4. Gómez, O.T., P.P.R. López, and J.S. Bacalla, *Criterios de selección de metodologías de desarrollo de software*. Industrial data, 2010. **13**(2): p. 70-74.
5. Duarte, A.O. and M. Rojas, *Las metodologías de desarrollo ágil como una oportunidad para la ingeniería del software educativo*. Revista Avances en Sistemas e Informática, 2008. **5**(2): p. 159-171.

6. Figueroa, M., *MeISE: Metodología de ingeniería de software educativo.* Revista Internacional Internacional Internacional de Educación en Ingeniería Educación en Ingeniería ISSN, 1940. **1116**.
7. Arias, M., Á. López, and J. Honmy, *Metodología dinámica para el desarrollo de software educativo.* 2015.
8. Alava, M.V., S.P. Delgado Figueroa, H.M. Blum Alcivar, and M.Y. Leyva Vazquez, *Single valued neutrosophic numbers and analytic hierarchy process for project selection.* Neutrosophic Sets and Systems, 2018. **21**(1): p. 13.
9. Teruel, K.P., J.C. Cedeno, H.L. Gavilanez, and C.B. Diaz, *A framework for selecting cloud computing services based on consensus under single valued neutrosophic numbers.* Neutrosophic Sets and Systems, 2018. **22**(1): p. 4.
10. Villamar, C.M., J. Suarez, L. Coloma, C. Vera, and M. Leyva, *Analysis of Technological Innovation Contribution to Gross Domestic Product Based on Neutrosophic Cognitive Maps and Neutrosophic Numbers.* Neutrosophic Sets and Systems, 2019. **30**(1): p. 3.
11. Gómez, G.Á. and J.E. Ricardo, *Método para medir la formación de competencias pedagógicas mediante números neutrosóficos de valor único.* Neutrosophic Computing and Machine Learning, 2020. **11**.
12. Ortega, R.G., M. Rodríguez, M.L. Vázquez, and J.E. Ricardo, *Pestel analysis based on neutrosophic cognitive maps and neutrosophic numbers for the sinos river basin management.* Neutrosophic Sets and Systems, 2019. **26**(1): p. 16.
13. Smarandache, F., J.E. Ricardo, E.G. Caballero, M.Y. Leyva Vázquez, and N.B. Hernández, *Delphi method for evaluating scientific research proposals in a neutrosophic environment.* Neutrosophic Sets & Systems, 2020. **34**.
14. Machiraju, S. and R. Modi, *Developing Bots with Microsoft Bots Framework.* 2018.
15. Tashchiyan, G.O., A.V. Sushko, and S.V. Grichin. *Microsoft Business Solutions-Axapta as a basis for automated monitoring of high technology products competitiveness*. in *IOP Conference Series: Materials Science and Engineering*. 2015. IOP Publishing.
16. Tascon, C. and H. Domínguez, *Análisis a la utilidad de la técnica de escenarios en la elicitación de requisitos.* Revista Antioqueña de las Ciencias Computacionales, 2017. **7**(1).
17. Jimeno Flores, J.V., C. Hernández, and G. Milckar, *Metodología para medir el nivel de rendimiento de los probadores de software en la empresa G & V Servigen SAC.* 2018.
18. Serna, E., R. Martínez, P. Tamayo, and I.U. de Envigado, *Una revisión a la realidad de la automatización de las pruebas del software.* Computación y Sistemas, 2019. **23**(1): p. 169-183.
19. Villada Zapata, C.C., C.A. Cifuentes Hincapie, V.M. Delgado Pena, and O.H. Rojas García, *Profundización de pruebas de software website MercadoLibre.* 2019.
20. Mar, O. and B. Bron, *Base Orientadora de la Acción para el desarrollo de prácticas en un Sistema de Laboratorios a Distancia* Revista Científica, 2017. **2**(29): p. 140-148.
21. Dávila, A., C. García, and S. Cóndor, *Análisis exploratorio en la adopción de prácticas de pruebas de software de la ISO/IEC 29119-2 en organizaciones de Lima, Perú.* RISTI-Revista Ibérica de Sistemas e Tecnologias de Informação, 2017(21): p. 1-17.
22. Grajales Quintero, A., E. Serrano Moya, and C. Hahan Von, *Los métodos y procesos multicriterio para la evaluación.* Luna Azul, 2013. **36**(1): p. 285-306.
23. Bouza, C. *Métodos cuantitativos para la toma de decisiones en contabilidad, administración, economía*. 2016; Available from: https://www.researchgate.net/publication/303551295_METODOS_CUANTITATIVOS_PARA_LA_TOMA_DE_DECISIONES_EN_CONTABILIDAD_ADMINISTRACION_ECONOMIA.
24. Ortega, R.G., M.L. Vazquez, J.A. Sganderla Figueiredo, and A. Guijarro-Rodriguez, *Sinos river basin social-environmental prospective assessment of water quality management using fuzzy cognitive maps and neutrosophic AHP-TOPSIS.* Neutrosophic Sets and Systems, 2018. **23**(1): p. 13.
25. Padilla, R.C., J.G. Ruiz, M.V. Alava, and M.L. Vázquez, *Modelo de recomendación basado en conocimiento empleando números SVN.* Neutrosophic Computing and Machine Learning, 2018. **1**: p. 31-36.
26. Smarandache, F., *Lógica neutrosófica refinada n-valuada y sus aplicaciones a la física.* Neutrosophics Computing and Machine Learning, 2018. **2**.
27. Le, C., *Preamble to Neutrosophy and Neutrosophic Logic.* MULTIPLE VALUED LOGIC, 2002. **8**(3): p. 285-296.
28. Smarandache, F., *Neutrosophy, a new Branch of Philosophy*. 2002: Infinite Study.
29. Villamar, C.M., J. Suarez, L.D.L. Coloma, C. Vera, and M. Leyva, *Analysis of technological innovation contribution to gross domestic product based on neutrosophic cognitive maps and neutrosophic numbers*. 2019: Infinite Study.
30. Smarandache, F. and M. Ali, *Neutrosophic Triplet Group (revisited).* Neutrosophic Sets and Systems, 2019. **26**(1): p. 2.
31. Zaied, A.N.H., A. Gamal, and M. Ismail, *An Integrated Neutrosophic and TOPSIS for Evaluating Airline Service Quality.* Neutrosophic Sets and Systems, 2019. **29**(1): p. 3.
32. Smarandache, F., *A Unifying Field in Logics: Neutrosophic Logic.* Philosophy, 1999: p. 1-141.
33. Wang, H., F. Smarandache, R. Sunderraman, and Y.Q. Zhang, *Interval Neutrosophic Sets and Logic: Theory and Applications in Computing: Theory and Applications in Computing*. 2005: Hexis.

34. Smarandache, F., M.A. Quiroz-Martínez, J.E. Ricardo, and N. Batista, *APPLICATION OF NEUTROSOPHIC OFFSETS FOR DIGITAL IMAGE PROCESSING.* Investigacion Operacional, 2020. **41**: p. 603-610.

35. Calzada, M.P. and M.C. Hernández, *Modelo de Recomendación Neutrosófico para el análisis Socio-Epidemiológico y funcionamiento familiar de pacientes alcohólicos.* Neutrosophic Computing and Machine Learning, 2020. **12**: p. 18.

36. Grimaldo Lorente, C.G., V. Hugo Lucero, M. Chulde, and J. Cadena, *A Model of neutrosophic recommendation for the improvement of the consents of the ICSID arbitration procedure in Bolivia, Ecuador and Venezuela.* Neutrosophic Sets & Systems, 2019. **26**.

Received: April 07, 2020. Accepted: August 09, 2020