Hybrid Method for Software Development Based On Agile Methodologies in the Coding and Testing Phases

Ing. Miguel Ángel Muñoz Pozos[1], Ing. Ayizdeth Fuentes Zarate[2], Dr. René Santaolaya Salgado[3], Dr José Juan Hernández Mora[4]

[1]Tecnológico Nacional de México/Apizaco, Carrera Apizaco-Tzompantepec. esquina con Av. Instituto Tecnológico s/n, Conurbado Apizaco-Tzompantepec, Tlaxcala, México C.P. 90300
M13370809[at]apizaco.tecnm.mx

[2]Tecnológico Nacional de México/Apizaco, Carrera Apizaco-Tzompantepec. esquina con Av. Instituto Tecnológico s/n, Conurbado Apizaco-Tzompantepec, Tlaxcala, México C.P. 90300
M13370774[at]apizaco.tecnm.mx

[3]Tecnológico Nacional de México/CENIDET, Interior Internado Palmira S/N, Palmira, 62490 Cuernavaca, Morelos, Mexico
rene.ss[at]cenidet.tecnm.mx

[4]Tecnológico Nacional de México/Apizaco, Carrera Apizaco-Tzompantepec. esquina con Av. Instituto Tecnológico s/n, Conurbado Apizaco-Tzompantepec, Tlaxcala, México C.P. 90300
juan.hm[at]apizaco.tecnm.mx

Abstract: ***This article proposes a hybrid method for the coding and testing phases in software development that involves both traditional methodologies and agile methodologies in the complete Development cycle. In the coding phase, agile techniques Scrum and extreme programming (XP) are used, taking the sprint section from scrum while extreme programming takes programming in pairs, resulting in a hybrid method (Scrum / XP). For the coding phase, this hybrid method takes the use of a version control flow very seriously, counting on the "git Flow" technique as the one recommended for software development. While in the testing phase the use of black box and white box tests is suggested, it is also recommended to use static and dynamic techniques. In this phase you can use "TDD or Test-Driven Development", in which tests are first and development second.***

Keywords: Agile method, coding phase, testing phase, version control

1. Introduction

Currently agile methods provide various benefits to software development, such as: incremental project deliveries, quick releases, flexible to change. However, one of its limitations is the lack of documentation in the different stages of software development. Since the 1990s, agile methods have been considered the most promising route to successful software development. A considerable number of publications of studies about agile methods present the benefits that companies obtain by adopting agile methods, as in the article *"Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practice"* [1], which makes a comparison between the various methodologies and, as a result, it is noted that Scrum has become the most popular agile method, in addition that agile methods are adapted and combined with other processes, and that companies are inclined towards the use of methods hybrids like the one proposed in this article. On the other hand, surveys on agile methods suggest that they are rarely used in their "pure" form. For example, in the article *"What Do Practitioners Vary in Using Scrum?"* [2], a study is published showing how Scrum is used in practice (with a particular focus on variations and the respective justification). Agile software development has become a popular way of developing software. Scrum is the most widely used agile framework, but it is reported that in practice it is often adapted.

The authors Tripp and Armstrong, in their article *"Exploring the Relationship between Organizational Adoption Motives and the Tailoring of Agile Methods"* They publish that in their study they found that there are three reasons for agile adoption: *a) the desire for higher quality software, b) greater efficiency or greater effectiveness, c) association with different configurations of agile practices, focused on project management; and agile practices related to the software development approach* [3]. They also show that the most popular agile methods are: Extreme Programming (XP), Scrum, Dynamic Systems Development Method (DSDM), Crystal and Function-based Development (FDD).

Among other things, the Tripp and Armstrong studies [3], they also show that traditional process models remain of some relevance. [4] Scrum and XP are the most popular methodologies adopted by the development community; they even find that the waterfall / XP and Scrum / XP combinations are the most common. Recently published studies by the authors M. Kuhrmann and D. Méndez Fernández [5], indicate a situation where traditional and agile approaches coexist, as practically (mixed / hybrid) approaches.

2. Hybrid method

Currently, agile methods provide various benefits to software development, such as: incremental project deliveries, quick releases, flexible to changes. However, one of its limitations is the lack of documentation.

The hybrid method proposed in this article consists of combining the advantages that SCRUM offers, such as: *the iterations with sprint, priority list and extreme programming methodology (XP) that during coding performs programming in pairs.* It also takes advantage of traditional methodologies such as technical documentation. In the stages of requirements gathering, *analysis and the definition of a coding standard* in which the Delphi method is applied.

A diagram outlining the proposed hybrid method is shown in Figure 1.

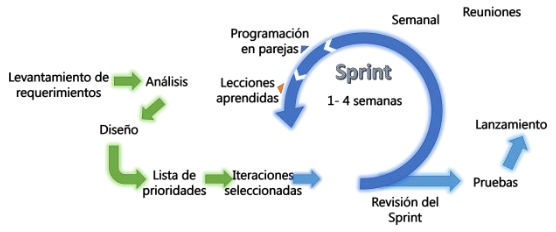

Figure 1: Hybrid method sketch

3. Coding Phase

Coding, or also called software programming, consists of transforming the requirements and the design into a defined programming language. This task is carried out by the team of programmers, following a coding standard. In this work this standard is defined by consensus using the Delphi method.

This method is a structured communication technique, developed as a systematic and interactive method of prediction, which is based on a panel of experts, where the cornerstone consists of a series of questionnaires defined by the group of participants within the procedure.

The method proposed in the coding phase is based mainly on Sprint, which is the coding of each of the user stories, given their level of priority and dependence on the others. Each Sprint is the creation of a piece of usable version software, which takes 1 to 4 weeks to build. The entrance to this phase is the requirements, objectives, design, work plan and the deliverable at the end of each Sprint.

In the method proposed in this article, a Sprint is made up of the following elements: *Selected iterations*, *weekly meetings, pair programming (using standard coding and versioning), lessons learned, and Sprint review*. Which are described below; Figure 2 shows the workflow of each Sprint in the coding phase.

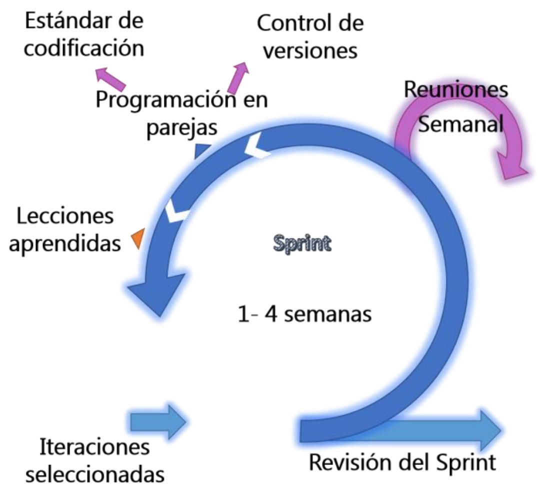

Figure 2: Coding

In this method, the *selected iterations* are the beginning of the planning of each sprint, and the user stories are selected, given their priority to begin with the coding of the solution for each one of them. Fulfilling the requirements and the design defined in the two previous phases.

The *weekly meetings* consist of meetings where the work team expresses what has been achieved since the last meeting, as well as the problems that have arisen. This is how you know when you already have a solution or to reach the right solution as quickly as possible with the help of the team. These meetings, mainly, should be done in less than an hour and present the functional software of the solution to the client or the project leader.

The *coding* is done with a technique used in agile software development, mainly in the extreme XP programming methodology, which consists of programming in pairs where two programmers work together, one of them being the driver, who is the one who writes the source code; while the other is the observer who inspects the driver's progress. Programmers alternate these roles. In this method, it is recommended that each programmer dedicate himself to solving a user story to provide greater results, more advances in less time, and that each programmer carry both roles with him.

A *coding standard* is a homogeneous programming style. In a project it allows all participants to understand it in less time, and consequently the code to be maintainable and durable. The agile extreme programming methodology emphasizes that programmers' communication is through code, making it essential that certain programming standards be followed to keep the code readable and maintainable during coding and software maintenance. It is for this reason that the proposed hybrid method defines a coding standard with the help of the Delphi method. Figure 3 shows the steps to follow to achieve a coding standard based on the knowledge and experience of experts in software development.

The *Delphi method* [6] is an information gathering technique that allows obtaining the opinion of a group of experts through repeated consultation. This qualitative technique is recommended when there is insufficient or necessary information for decision-making. This article describes the main characteristics of the technique and details the process of repeated consultations in the application of the technique.

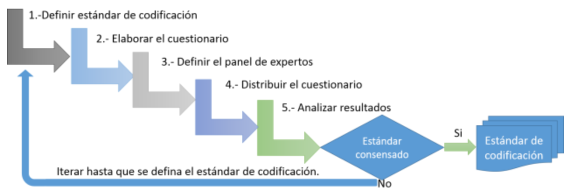
1.-Definir estándar de codificación
2.- Elaborar el cuestionario
3.- Definir el panel de expertos
4.- Distribuir el cuestionario
5.- Analizar resultados
Estándar consensado
Si
Estándar de codificación
Iterar hasta que se defina el estándar de codificación.
No

Figure 3: Delphi method to define encoding standard

Version control is a system that records changes made to a file or set of files over time, so that later, specific versions can be recovered from earlier [7]. This is why, in the proposed hybrid method, it is necessary to have a version control scheme in the coding phase, for the benefit of programmers; but mainly, to demonstrate the growth of the project, during the Sprint and its culmination. Some other reasons why you should have a version control system are:

a. Know who has been responsible for a certain modification and when it was made
b. Make comparisons between versions of an application
c. Observe the evolution of the project over time
d. Be aware of source code changes
e. Have a backup of the project
f. Have a history in which the modifications made to the code are detailed

Figure 4 shows the work cycle to be followed for the use of a version control system for the hybrid method and for projects that require the use of a version control system. It is made up of three areas *working directory, staging area, and git repository.*

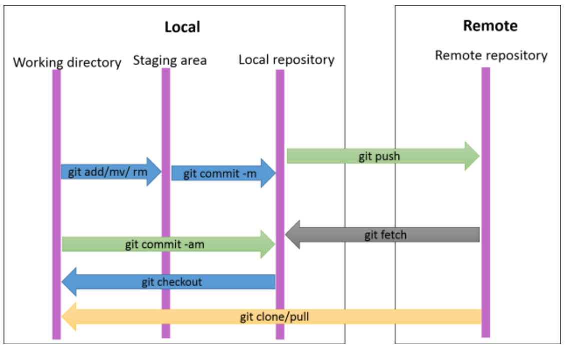

Figure 4: The lifecycle of file status

When you are a Version Control System, there are workflows with production branches that are:

Centralized workflow: This is a great Git workflow for teams migrating from Apache Subversion. Like Subversion, the centralized workflow uses a central repository as the entry point for all changes to the project. Instead of *"trunk"*, the default development branch is named "master", and all changes are committed to that branch. This workflow does not require more branches than "*master*".

Figure 5 shows how the branch and the version of the project are structured using this technique for the development of the work.

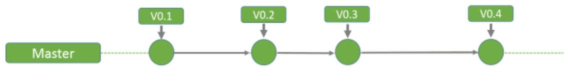

Figure 5: Basic branching

Feature branch workflow: The main idea behind the feature branch workflow is that development of a feature should be done in a specialized branch, rather than a master branch. This isolation allows multiple developers to work on a specific feature without disturbing the content of the main codebase. It also implies that the *"master"* branch will in no case contain erroneous code, which is a great advantage for continuous integration environments. Figure 6 shows how the branches are organized with this work option.

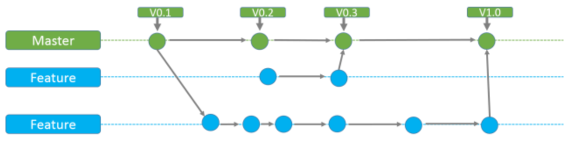

Figure 6: Characteristic branches

Gitflow workflow: Ideal for projects that have a scheduled release cycle. This workflow does not add any concepts or new commands beyond what is needed for the feature branch workflow. Instead, it assigns specific roles to the different branches and defines how and when they should interact. In addition to feature branches, individual branches are used to prepare, maintain, and register publications. Of course, you also take advantage of the benefits of the feature branch workflow, such as: pull requests, isolated experiments, and more efficient collaboration. In Figure 7 you can see how branches are integrated to handle software development more efficiently.

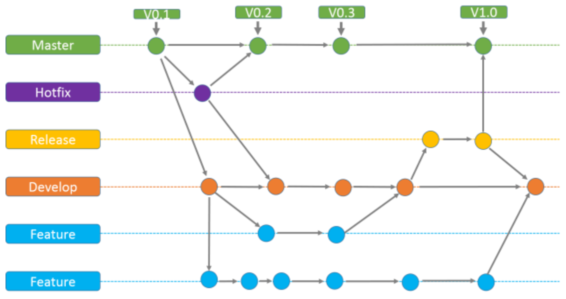

Figure 7: Git Flow

The *lessons learned* from the process are generally collected at the end of each Sprint, to determine and analyze the elements of each user story that were successful or not. For the project team to learn from experiences in product development. These lessons are composed of the knowledge acquired through experience. It reflects information on successes or failures and is a valuable source of information for future projects.

The *Sprint Review* is the step where the team reviews that what is requested in the User Stories has been completed and performed correctly. If the user stories are approved, it goes to the testing phase. This is done to see if the product or user story is finished and can go through testing without any missing or incomplete features.

4. Test phase

It is in the testing phase of the proposed hybrid method that each user story is passed through some type of evaluation, see Figure 8. This phase is supported by some techniques to be able to perform them efficiently to locate and reduce defects in the applications under development; some of these techniques are shown in Figure 9.

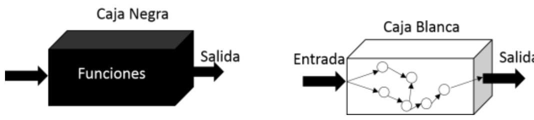

Figure 8: Types of software tests

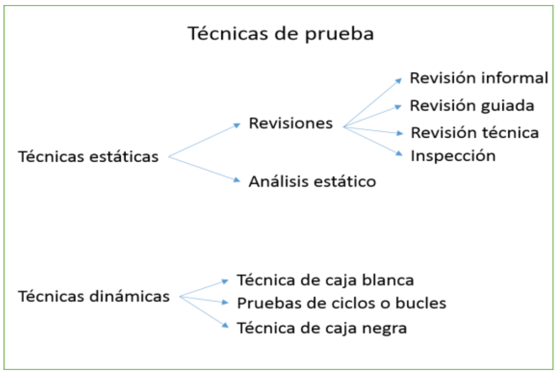

Figure 9: Testing techniques

Dynamic testing is the process of running a program with the intention of finding defects [8].

Some of the reasons why any software should be tested are:

a. Detect as many defects as possible
b. Help managers with decision making
c. Find the safe scenarios for the use of the product
d. Assess quality
e. Check product correctness
f. Quality assurance
g. Commercial competitiveness
h. Achieve higher quality in software
i. Changes in technology
j. Cost and risk reduction
k. Increased productivity

And the main objective of the tests is to bring quality to the product that is being developed.

To find the defects in the code, two of the techniques most used in testing can be used are the dynamic techniques of "white box" and "black box" see Figure 9. Although there are tools such as Test-guided Development (TDD) See Figure 10 where we can see how this tool can be used to test each requirement, and thus cover the testing part of the software in question.

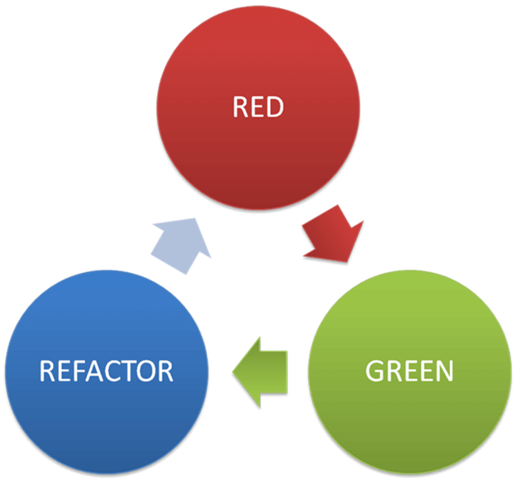

Figure 10: The TDD cycle

It should be noted that in software development, the testing phase is critical to ensure that the product is sent to the production environment with the quality expected by the customer. That is why today it is essential to have a Software Testing Plan to carefully specify the functions to be tested, how the tests will be executed, who will be responsible and the schedule for their execution. The informatics project office (pmoinformatica) [9] has many templates for multiple processes. One of these is for a software test plan, where a template based on pmoinformatics is found, to achieve the definition of a software test plan.

5. Evaluation and results of the practice of the proposed hybrid method

The developed method was subjected to two evaluations. The first was applying the evaluation approach shown in Figure 1, the second was applying the method to a real case. This hybrid method was tested in the migration of an information system called "teacher evaluation" for the technological institute of Apizaco, and excellent results were obtained. Table 1 shows a comparison between using a method for software development.

Table 1: Method comparison

	Teacher evaluation	
Questions	Created without methodology	Created with "The hybrid method"
Documentation	Does not apply	Wide and durable
Stages of development	Not defined	• Survey of requirements • Requirements analysis • Software design • Coding • Tests • Launching
Agility	Not defined	High
Durability	Not defined	High
Scalability	Not defined	High

6. Conclusions

The proposed hybrid method for the coding and testing phase is intended to cover the limitations of agile methodologies, which is the lack of documentation. In this article, each activity of the coding and testing phases was described, as well as the proposal detailing each phase of how the application of each of them would be. As we have seen, several authors are inclined to believe that hybrid methods for software development are the best solution, which is why the method proposed in this article integrates various software development process models, taking and combining the capabilities or advantages relevant to each of them.

References

[1] G. Theocharis, M. Kuhrmann, J. Münch, P. Diebold. "Is Water-Scrum-Fall Reality? On the Use of Agile and Traditional Development Practices". Lecture Notes in Computer Science. 9459, pp. 149-166, 2015.
[2] P. Diebold, J. Ostberg, S. Wagner, U. Zendler. "What Do Practitioners Vary in Using Scrum?". Lecture Notes in Business Information Processing. 212, pp. 40-51, 2015.
[3] J. F. Tripp and D. J. Armstrong, "Exploring the Relationship between Organizational Adoption Motives and the Tailoring of Agile Methods," 2014 47th Hawaii International Conference on System Sciences, Waikoloa, HI, pp. 4799-4806, 2014.
[4] A. Solinski, K. Petersen, "Prioritizing agile benefits and limitations in relation to practice usage". Software Qual J 24, pp. 447-482, 2016.
[5] M. Kuhrmann y D. Méndez Fernández, "Systematic Software Development: A State of the Practice Report from Germany". 2015
[6] M. Reguant Alvarez, M. Torrado Fonseca. "El método Delphi", Reire. 9, pp. 87-102, 2016.
[7] S. Chacon, "Getting Started About Version Control". May, 09, 2014, [Online]. Available: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control. [Accessed: feb. 12, 2020].
[8] G. J. Myers and S. Corey, The Art of Software Testing, USA, 2004.
[9] R. A. Rodríguez Morillo, "La oficina de proyectos de informática" www.pmoinformatica.com, Jan. 31, 2001. [Online]. Available: http://www.pmoinformatica.com/p/plantillas-de-gerencia-de-proyectos.htm [Accessed: Enero. 12, 2020].

Author Profile

Miguel Angel Muñoz Pozos received the degree of Eng. In Information and Communication Technologies from Tecnologico de Apizaco in 2018, during 2013-2018 he belonged to the degree in Information and Communication Technologies, from August 2018 to date he is In the division of postgraduate studies, he completed a master's degree in computer systems.

Ayizdeth Fuentes Zarate received the B.I. degree in Engineering in information and communication technologies from Technological Institute of Apizaco in 2018. During 2016-2019, she stayed in Computer Center from the same institute to develop Web-based information systems to use of students, teachers and administrative staff. She now studies Master's degree in Computer Systems.

René Santaolaya Salgado is professor of Technological Research and Develop National Center (CENIDET), Cuernavaca, Morelos, México. His research interest includes software process model, software architecture, software reuse and web services. René Santaolaya received a PhD in Computer Science (Software Engineering) from National Polytechnic Institute, Computing Research Center. He is a senior member of the IEEE. He has been developer, analyst and project leader in the Academic Body of Software Engineer.

José Juan Hernández Mora has a Doctor of Teaching Excellence degree from the University of Los Angeles, 2019. He also has a degree in Computer engineer from the Universidad Autónoma de Tlaxcala, from 1994. Master's in computer science at the National Center for Research and Technological Development of the TecNM, 2003. Research professor at the Tecnológico de Apizaco del TecNM. Teacher of the Master of computer systems of the TecNM / Instituto Tecnológico de Apizaco.