DeSoftIn: A methodological proposal for individual software development

DeSoftIn: Propuesta metodológica para desarrollo de software individual

DeSoftIn: Proposta metodológica para desenvolvimento de software individual

Juan Sebastián González-Sanabria*
Juan Antonio Morente-Molinera**
Alexander Castro-Romero***

Fecha de recepción: 21 de febrero de 2017
Fecha de aprobación: 17 de abril de 2017

Abstract

Different Computer Engineering undergraduate programs over the world are demanding the students to present individual works and, particularly, to present a degree project, which in most cases is related to software development. However, when planning the projects, students find themselves with the problem of choosing a method to develop the software, since existent methods involve team work, but the degree project is supposed to be done individually, in order to evaluate the student's acquired knowledge. This difficulty leads to projects that fail either to achieve the proposed objectives or to finish on the expected time, among other difficulties. This paper presents a methodological proposal for the development of individual software projects, mainly in academia, named "DeSoftIn", which will contribute to accomplish the project objectives, and will allow the students to approach development methodologies since the beginning of their studies.

Keywords: Agile methods; Development methodology; Software quality; Software engineering.

Resumen

Los diferentes programas de pregrado de Ingeniería Informática en el mundo, exigen a sus estudiantes presentar trabajos de manera individual y, particularmente, un proyecto de trabajo de grado, los cuales, en la mayoría de los casos, están relacionados con el desarrollo de un software; sin embargo, al momento de planear dichos proyectos, los estudiantes se encuentran ante la dificultad de escoger qué metodología utilizar, pues las metodologías de desarrollo de software existentes suponen grupos de personas, y resulta que con el fin de evaluar los conocimientos particulares adquiridos por cada estudiante, los trabajos de grado se deben hacer, generalmente, de manera individual. La dificultad en la selección de la metodología lleva a que los proyectos no den como resultado el objetivo propuesto o tarden más de lo programado, entre otras dificultades. El presente artículo plantea una

* M. Sc. (c) Universidad Pedagógica y Tecnológica de Colombia (Tunja-Boyacá, Colombia). juansebastian.gonzalez@uptc.edu.co. ORCID: 0000-0002-1024-6077.
** Ph. D. Universidad Internacional de La Rioja (La Rioja, España). jamoren@ decsai.ugr.es. ORCID: 0000-0002-2729-6900.
*** M. Sc. Universidad Pedagógica y Tecnológica de Colombia (Tunja-Boyacá, Colombia). alexander.castro01@uptc.edu.co. ORCID: 0000-0001-9469-5445.

propuesta metodológica para el desarrollo individual de proyectos de software, principalmente en la academia, denominado DeSoftIn, que coadyuve al cumplimiento de los objetivos del proyecto y permita a los estudiantes tener una aproximación al uso de metodologías de desarrollo, desde el inicio del programa de estudios.

Palabras clave: Calidad de software; Ingeniería de software; Metodologías ágiles; Metodologías de desarrollo.

Resumo

Os diferentes programas de graduação de Engenharia Informática no mundo, exigem a seus estudantes apresentar trabalhos de maneira individual e, particularmente, um projeto de trabalho de graduação, os quais, na maioria dos casos, estão relacionados com o desenvolvimento de um software; porém, ao momento de planejar tais projetos, os estudantes encontram-se perante a dificuldade de escolher qual metodologia utilizar, pois as metodologias de desenvolvimento de software existentes supõem grupos de pessoas, e acontece que para avaliar os conhecimentos particulares adquiridos por cada estudante, os trabalhos de graduação devem ser feitos, geralmente, de forma individual. A dificuldade na seleção da metodologia faz com que os projetos não deem como resultado o objetivo proposto ou tardem mais do que foi programado, entre outras dificuldades. O presente artigo planteia uma proposta metodológica para o desenvolvimento individual de projetos de software, principalmente na academia, denominado DeSoftIn, que contribua ao cumprimento dos objetivos do projeto e permita aos estudantes ter uma aproximação ao uso de metodologias de desenvolvimento, desde o início do programa de estudos.

Palavras chave: Qualidade de software; Engenharia de software; Metodologias ágeis; Metodologias de desenvolvimento.

Cómo citar este artículo:
J. S. Gónzalez-Sanabria, J. A. Morente-Molinera, and A. Castro-Romero, "DeSoftIn: A methodological proposal for individual software development," *Rev. Fac. Ing.*, vol. 26 (44), pp. 23-32, May. 2017

I. Introduction

Since the appearance of computer engineering as an academic discipline, models, frames and methodologies that describe the "basic steps" or ideals to adequately carry out software development projects have been proposed. Nevertheless, the lack of homogeneity on factors such as development styles, working teams, and resources, among others**,** have resulted in the existence of many methodologies, mainly focused on team work (*i.e.,* two or more people). Additionally, the students interested in this discipline, during their academic formation, are continually compelled to conduct individual projects, mostly without orientation or a methodology. Therefore, during this process, the students continuously encounter problems and make mistakes that are not detected until the resulting product is turned in.

Current methodologies used to develop software (eXtreme Programming –XP, Cascade, and Iteractiv, among others) propose the conformation of teams with at least 5 people, which constitutes the major difficulty to apply them to individual projects. Moreover, in these methodologies, each team person complies with very specific functions, and in several phases there are not transversal communications among them. On the other hand, the delivery time is usually 15-30 days per delivery, which means an estimated of 90-120 days to have the final product; however, students do not have that long in an academic project, since they are given only 30 to 60 days to complete these projects. Furthermore, a significant investment in financial and physical resources is required, which is not taken into account in most of the software development academic projects.

Based on expert opinions, it can be deduced that an individual software development methodology must count with all the quality and efficiency aspects of a product developed by a team. Such methodology should provide the necessary phases and tools to offer the versatility of the traditionally used methodologies, complying with deadlines, objectives, and defined scopes of the project.

With this motivation, in this paper, we formulate a methodological proposal to develop software, called "DeSoftIn", which allows, mainly computer engineering students, to have a reference point when they have to work individually. To achieve this, initially, we searched for current theories, and compared the most used methodologies, which is explained in the next chapter; subsequent chapters describe the developed methodological proposal, the evaluation of such proposal, and the conclusions and recommendations.

II. Methodology

To develop this research, we defined three main phases: 1) literature search to elaborate the state of the art of the research topic; 2) selection and comparison of the current most used methodologies, based on widely known cases where they have been used, plus experiences of professionals in the area; and 3) proposal of a methodology to individually develop software, application of such methodology in a study case, and evaluation using the 4-DAT (4-Dimensional Analytical Tool) method.

In the first phase, we carried out a systematic study on the investigations conducted during the past five years, focused on software development methodologies and their applications in different contexts; for this, we searched articles in different indexes, indicators, and scientific data bases, such as Re0dalyc and IEEE Xplore. After gathering the articles, we read those that had more citations, with the objective to select the ones most closely related to the objective of our study. The most relevant selected articles are presented in Table 1.

Table 1
Literature Revision

Title	Year	Reference
Propuesta pedagógica: Una metodología de desarrollo de software para la enseñanza universitaria	2011	[2]
Propuesta metodológica para desarrollo de software educativo en la Universidad de Holguín	2016	[3]
Taxonomía de los modelos y metodologías de desarrollo de software más utilizados	2012	[4]
Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje –MESOVA–	2011	[5]
Metodología ágil para equipos pequeños usando plataformas Microsoft	2011	[6]
Análisis de alternativas metodológicas para el desarrollo de software educativo	2014	[7]
El desarrollo de software dirigido por modelos en los repositorios institucionales	2014	[8]
Proceso para gestionar riesgos en proyectos de desarrollo de software	2013	[9]
Reflexiones acerca de la adopción de enfoques centrados en modelos en el desarrollo de software	2011	[10]

In the second phase, we selected and categorized the software development methodologies, particularly those known as agile, searching in the selected literature and interviewing professionals in the area; this with the goal of establishing a comparison that would allow us to detect failures in those methodologies when applied to the projects individually developed.

Once we identified the advantages and disadvantages that the selected developing methodologies have when applied to individual projects, we extracted their most relevant characteristics, and those with the best reception among developers; thanks to this, we were able to propose a methodology supported by experiences, successful cases, and expert opinions. This methodological proposal was applied in a study case, with the aim to evaluate its performance in relation to methodologies with longer trajectory and success; additionally, it was evaluated by the 4-DAT method.

III. Methodological proposal "DeSoftIn"

In this section, we formulate the methodological proposal, explaining the necessaries phases, roles, abilities, and skills necessary to successfully accomplish the individual projects.

A. Phases

1) Planning and analysis: The main action to be accomplished in this phase is the definition of the project scope, which should be accompanied with the analysis of requirements, in order to establish or estimate times, as well as to evaluate the required knowledge on tools, technics, and technologies to be used.

In order to plan and control the time, it is important to differentiate in the analysis between what it should be done and what can be achieved, since it is necessary to take into account the customer limitations and restrictions, mainly at the resources level. Once this is clear, the activities that will be carried out in each one of the sprints are defined, according to their prioritization. These activities will be represented on a timeline that, taking into account the "Last Planner System" [11], is revised backwards from end to beginning, with the goal of supplying and disposing the required resources beforehand, and thus, avoiding waiting until the last minute to search for such resources. Posteriorly, a deadline to complete the development must be set, which should include the necessary time to get qualified or to learn any of the required aspects that are not yet mastered by the developer.

It is important to include, like in any development, risk planning, in order to identify those responsible to apply the defined answers for each risk during the application development. Lastly, the requirements

should be a priority, so that the design can begin with those that are the most important and transversal to the whole project; this will allow to have, after the first delivery, a functional product.

It should be pointed out that the requirements, contrary to the proposal for the documentation of the rest of the methodologies, are not presented with the traditional forms; instead, we propose to use a checklist of the customer required functionalities, linking them to the system roles (Fig 1). In the checklist, those users that intervene in a specific functionality are marked with an X, which simplifies for the developer the unified control over the permissions, roles, and functionalities of the project.

	Role 1	Role 2	…	Role n
Requirement 1		X		X
Requirement 2	X			
…				
Requirement n	X			

Fig. 1. Checklist for requirement gathering.

The planning and analysis phase is not compulsory taken as incremental, due that, by nature, the academic projects determine from the beginning all the requirements. In case of including new requirements during the project development, it is suggested to end the initially defined requirements, and then to include an iteration of this phase to introduce them.

2) Design: Once the requirements are defined (previous phase), they are gradually included in each design delivery, according to their priorities. Also, in this phase, the necessary information for the optimum implementation of the requirements must be compiled and complemented.

To make the different diagrams, we suggest to use the BPMN (Business Process Model and Notation); nevertheless, it is left to the designer's judgment the elaboration of a diagram that provides a global vision of the business. Additionally, the interphase prototypes should be made in this phase, so they can be validated with the customer, and can be approved and improved to pass to the next phase.

3) Development: The development phase is the one that implies the major responsibilities within the interactive phases, because in this one, the approved requirements should be "coded", in order to obtain a functional result; likewise, in this phase the developer's self-criticism and expertise are tested.

After the first delivery, or first sprint if it gets associated with SCRUM, additional recommendations made by the consultant or the client, as well as the requirements to be developed in each one of the deliveries are included. Both in the design and in the development, the form shown in figure 1 can be used, including a color range that indicates the progress of the fulfillment of each requirement. In figure 2, the color green indicates a developed requirement that has been approved by the client; orange, a requirement that is in evaluation phase; yellow, a requirement that is in development; and red, those functionalities that have not yet been initiated.

	Role 1	Role 2	…	Role n
Requirement 1				
Requirement 2				
…				
Requirement n				

Fig. 2. Requirement advance control.

4) Implementation: Once development is complete, the prototype must be applied, validated, and tested. In these three processes, attention should be paid to the recommendations and suggestions found in norms and standards on software quality models, such as ISO/IEC 15504 [12]. Likewise, regarding the testing process, guidance from standards and norms on information security, such as the ISO 27000, is suggested.

Additionally, in this phase, the functionalities developed at each delivery must be integrated, and the respective quality and integration tests must be carried out. It is important to include in this phase the issues related with risk management, in order to establish monitoring and control strategies. Also, the impact of every established risk should be taken into account, which requires great ability and knowledge from the developer, along with notable communication skills to alert the customer about such risks without causing alarm.

On the other hand, in case the developed software needs to be integrated into other system, the corresponding integration tests should be conducted during each

one of the iterations, with their respective developed functionalities.

5) Evaluation: At the end of each phase, which should not take longer than 10 days, a joint evaluation between the developer "team" and the customer should be carried out, in order to evaluate whether the developed and implemented product complies with the planned. Also, a meeting with the consultant or adviser is suggested, with the aim to obtain opinions, from a technical point of view, about the quality of the product that will be turned in. Figure 3 shows the flow among the described phases.

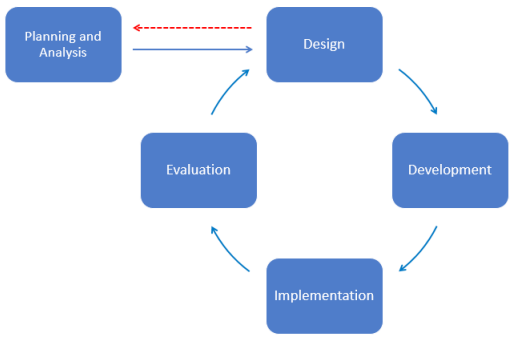

Fig. 2. Phase flow of "DeSoftIn".

B. Roles

Given that the projects are developed individually, it is not clear to talk about the roles; nevertheless, two roles are proposed within this methodology: the project team (a one-person team) and the consultant.

1) Project equipment: In the development of the methodology proposed here, the project team is composed by one person, who is responsible for all the different proposed duties, as the project leader, developer, tester, and architect, among others; additionally, he/she has the responsibility to decide how to organize the fulfillment of the proposed objectives for each iteration.

2) Consultant: Although the methodology is proposed as individual, it is suggested to have an external collaborator with specific knowledge on the project subject. This is included because the methodological proposal is oriented, firstly to undergraduate developments, in which the consultant labor can be performed by the student's advisor. Such consultant may contribute with ideas and experiences that would enrich the product, and since his/her work is less active, the dedication need is minimum.

C. Abilities and dexterities

This aspect presents the major difficulty for the methodological proposal because the same person should have too many abilities, both technical and personal. Among the main abilities that should stand out in those persons applying the methodology are the following:

- Active and patient communications skills to be able to abstract and retain the information given by the client during the requirements phase. Also, the person should be able to explain every decision made throughout the process, in order to clearly explain the obtained results.
- Ability to work individually without supervision, since the person should be able to challenge himself and control his own time, which demands a high level of discipline.
- Excellent formation and knowledge regarding the tools and technics to be used. Furthermore, the person should have a quick and effective learning capacity or "curve", in case there is a need for making adjustments or changes in a technology that was not contemplated at the beginning of the project.
- Knowledge on tests and software quality and safety tests. This ability is key in the methodological proposal, and the difficulty in its application consists in that is the same developer who conducts most of the evaluation to find errors and failures, both at code and functionalities levels, which may generate personal conflicts of interest.
- Capacity to determine, manage, and control risks in different environments, including, at the code level, natural disasters, and attacks, among others.

D. Devices and tecniques

Below, we present the techniques and devices that complement the forms presented in figures 1 and 2.

1) Sprints from 3 to 10 days: By using short sprints, it is possible to make quicker changes, and to have at the beginning small functional versions that allow the customer to have an idea of the product's direction. Additionally, the delivery time is defined in this lapse, since the time limit is shorter for individual

developments in the academy (less than two months). It is convenient and relevant to use continuous deliveries, to allow a permanent control by either the advisor or the customer.

2) Breaks between Sprints: Taking into account that the development will be carried out by only one person, it is convenient to define breaks between deliveries; this with the goal of giving some rest to the developer, so he/she can neutrally look at the development in process, and does not get overloaded with responsibilities. In these pauses, it is suggested to include or alternate the resting time with courses that allow acquiring new abilities to apply in the project.

3) Logbook: The developer should have a diary or logbook with the tasks that are being completed in the application. Because the developer is not interacting with other people, a logbook is the best way of efficiently control the changes and inclusions that are being done in the project, in addition to be the best way to evaluate the actions and fulfillment of tasks throughout the project flow.

4) CRC Cards: The CRC cards (Class-Responsibility-Collaborator) allow to control the assignment of responsibilities and the collaboration with other objects. Usually, there is one card per class, which summarizes the class responsibilities and the list of objects with which it collaborates to function [13].

5) Meetings: Once a deliver is finalized, first, a meeting with the customer should take place, followed by a meeting with the consultant, in order to either finish the sprint or plan the necessary adjustments. In both the meetings and the project planning, the dedication should be estimated.

6) Estimation of the dedication: Given that both roles, project leader and developer, fall under the responsibility of the same person, she/he is subject to an elevated level of discipline. Therefore, she/he should clearly differentiate between available work hours, and dedication hours, being the main difference between the two, the leisure hours. For this reason, it is suggested to take into account the formula (1) proposed in [13]:

$$V_E = D_{HD} * F_D \quad (1)$$

Where D_{HD} is the available days-man; F_D, the dedication factor; and V_E, the estimated advance speed of the project. The dedication factor is an estimation; in the case of individual development, it refers to the concentration level of the project developer; if this factor is low, it means that the person is susceptible to distractions and impediments (including familiar and personal distractions, among others) that would delay the project delivery time.

E. Tools

Despite most methodologies and metholodological proposals suggest the use of particular tools to control and manage the project, the present methodological proposal leaves this to the developer's own judgment; this with the goal of avoiding any bias in his criterion, and allowing him/her to use the tools he/she is already familiar with and feels more comfortable using, hence preventing him/her to invest time in learning new tools.

IV. Evaluation of "DeSoftIn" with 4-DAT

The 4-DAT method evaluates, among other aspects, whether a software developing methodology has taken into account the principles of the agile manifesto, in other words, whether it prioritizes people, has a communication orientation, is flexible (easy adaptability), fast (quick and iterative with functional versions of the product), efficient (short time and good quality), adaptable (proper reaction to changes), and has learning capacities (it can be improved during and after the development) [14].

Table 2
Evaluation of DeSoftIn with 4-DAT

"DeSoftIn"	FY	SD	LS	LG	RS	Total
(i) Phases						
Planning and Analysis	1	1	1	1	0	4
Design	1	1	1	1	1	5
Development	1	1	0	1	1	3
Application	1	0	0	1	1	4
Evaluation	1	0	0	1	1	4
Total	5	3	2	5	4	19
Agility Grade (AG)	5/5	3/5	2/5	5/5	4/5	19/25
(ii) Practices						
Interactive and incremental development by short iterations	1	1	1	1	1	5
Coupling	0	0	0	0	0	0
Tests	1	1	0	1	1	4
40 weekly hours	0	0	0	0	0	0
Quick feedback	0	1	0	1	1	3
Simple design	1	1	0	1	0	3
Refactoring	1	1	1	1	1	5
Active participation of all project members	1	1	0	1	1	4
Constant reuse	1	1	0	1	1	4
Programing style	1	0	0	1	1	3
Meeting to evaluate the finished iteration and to plan de next one	1	1	1	1	0	4
Progress report	1	1	1	1	1	5
Customer availability	1	1	1	1	1	5
Use of exclusive devices	1	1	1	1	1	5
Total	11	11	6	12	10	50
Agility Grade (SD)	11/14	11/14	6/14	12/14	10/14	50/70

The analysis of the results showed that one of the failures of the methodological proposal corresponds to the efficiency, underlining that the study case presented the most pessimistic values in this regard, that is, assuming that the person who will develop the project needs to learn some of the tools that will be used for it. Furthermore, when averaging the grades presented in Table 2, we obtained the values shown in in Table 3. Finally, Table 4 shows the obtained values when comparing the methodology proposal, XP, and Scrum [15].

Table 3
Average grades for DeSoftIn

	Accomplished characteristics	Average
Phases	19/25	0.76
Practices	50/70	0.71
Average		0.74

Table 4
COMPARISON OF XP, SCRUM AND DESOFTIN

	DeSoftIn	XP	SCRUM
Phases	0.76	0.70	0.60
Practices	0.71	0.73	0.80
Average	0.74	0.72	0.70

Taking into account the results shown in Table 4, even if DeSoftIn obtains a better grade than SCRUM, it is convenient to reiterate that this is a framework, which implies a low grade in the phases' part. Regarding the practices, SCRUM excels notably, and XP is superior to DeSoftIn, which implies that the proposed projected practices should be included or improved.

Although the grade obtained for the methodological proposal, based on dimension 2 of the 4-DAT method, is high, it is noticeable the low efficiency level obtained, which is due to the pessimistic panorama chosen when evaluating the methodology. These faults are complemented by the lack of control provided by an academic peer, and the need of higher dedication and commitment from only one person.

V. Conclusions

Despite we only present here a software development methodological proposal, and are aware it still needs to be tested under different environments to be improved, we can conclude the following regarding the "DeSoftIn" application:

- Minus is more: This is one of the proposal premises, due that it is based on having only the necessary documentation, without the use of the rigid and unnecessary forms, in some cases.

  The use of traditional methodologies and models is becoming inefficient due to small work teams, or the work at small scale, particularly because of the amount of forms, designs, and other artifacts that require a lot of time for their elaboration.

  Additionally, the use of complex, or even technical, language occasionally hampers the documentation understanding by third parties or by people in the team who did not participate in its writing. Therefore, the use of a logbook that records all the actions that take place in the project will allow a more simplified control.

- Quantity is not quality: In reference to the development teams, having big teams hinders communication, whereas within small teams, comprehension and understanding among its members is better. Therefore, although it could be risky that this proposal relies on only one person, when considering individual projects, this person tends to show more responsibility and personal commitment; additionally, in this methodology, we suggest the recurrent advise of an expert, which guides the developer about the actions that must be taken.

- Does the client always have the reason? Despite this is a well-known concept within the commercial circle, in the software development area, experts and professionals suggest the contrary: most of the customers do not really know what they want, and therefore, it is necessary to explain them what is possible, and help them to be realistic with their expectations.

  DeSoftIn answers to this necessity, and because it does not include the Planning and Analysis phase within the development cycle, it avoids the occurrence of abrupt changes in the requirements; nevertheless, it is possible to consider the client's opinions and criteria on the functionality designs obtained in the requirement analysis.

- Continuous learning: When the technologies and tools are imposed by the client, and the developer does not know them, it is necessary the decisive interest of the developer to learn them quickly; this, besides helping him to widen his knowledge on tools and techniques, encourages the continuous practice and update when no projects are under development.

- Development of qualities: One of the most criticized factors in the computation area is their professionals' insensibility; nevertheless, this methodological proposal achieves a great sense of responsibility and commitment in the developer, due that his/her reputation "is in play", which additionally allows to value the team work.

- Prioritization of simple and short functionalities, and continuous deliveries: This premise gives the developer a sense of increasing productivity, and allows a constant approximation between the client and the system, facilitating the client's contributions

and opinions that will prompt the success of the final product.

- Risks reduction: When the deliveries are far apart, the difference between the client's expectations and the developed product may be quite distant. Therefore, when the deliveries are set to be closer in time, the product can be realigned in real time, and therefore, the control and management of risks can be agilely determined. Although short times are an advantage in all agile methodologies, using continuous delivery practices reduce the time from 15 to 5 days, and even less, under the throwing modality.

This paper contributes to emphasize that DeSoftIn may become an academic formation element for computing students, given that, contrary to the rest of the methodologies, models and frameworks proposed for software development, it is centered in an academic context.

References

[1] J. P. Gamboa, and A. Rosado, "Diseño de un método ágil de desarrollo de software basado en XP, SCRUM, OPENUP y validado con la herramienta de análisis 4-DAT," *Ingenio*, vol. 8, pp. 19-31, 2015.

[2] E. Gottberg, G. Noguera, and M. A. Noguera, "Propuesta pedagógica: Una metodología de desarrollo de software para la enseñanza universitaria," *Universidades*, vol. LXI (50), pp. 49-57, 2011.

[3] C. J. Madariaga-Fernández, Y. Rivero-Peña, and A. R. Leyva-Téllez, "Propuesta metodológica para desarrollo de software educativo en la Universidad de Holguín," *Ciencias Holguín*, vol. XXII (4), pp. 1-17, 2016.

[4] J. Cervantes, and M. C. Gómez, "Taxonomía de los modelos y metodologías de desarrollo de software más utilizados," *Universidades,* vol. LXII (52), pp. 37-47, 2012.

[5] E. Parra, "Propuesta de metodología de desarrollo de software para objetos virtuales de aprendizaje -MESOVA," *Revista Virtual Universidad Católica del Norte*, nº 34, pp. 113-137, 2011.

[6] A. F. Loboguerrero, L. Castañeda Bueno, and H. F. Arboleda, "Metodología Ágil para equipos pequeños usando plataformas Microsoft," *Sistemas & Telemática,* vol. 9 (18), pp. 83-99, Sep. 2011. DOI: http://doi.org/10.18046/syt.v9i18.1078.

[7] I. Marcano, and G. Benigni, "Análisis de alternativas metodológicas para el desarrollo de software educativo," *SABER - Revista Multidisciplinaria del Consejo de Investigación de la Universidad de Oriente*, vol. 26 (3), pp. 297-304, 2014.

[8] J. Texier, M. De Guisti, and S. Gordillo, "El desarrollo de software dirigido por modelos en los repositorios institucionales," *Dyna*, vol. 81 (184), pp. 186-192, Apr. 2014. DOI: http://doi.org/10.15446/dyna.v81n184.37164.

[9] O. Pérez, and Y. Zulueta, "Proceso para gestionar riesgos en proyectos de desarrollo de software," *Revista Cubana de Ciencias Informáticas*, vol. 7 (2), pp. 206-221, 2013.

[10] J. Bernardo-Quintero, and J. F. Duitama-Muñoz, "Reflexiones acerca de la adopción de enfoques centrados en modelos en el desarrollo de software," *Ingeniería y Universidad,* vol. 15 (1), pp. 219-243, 2011.

[11] L. F. Botero, and M. E. Álvarez, "Last planner, un avance en la planificación y control de proyectos de construcción Estudio del caso de la ciudad de Medellín," *Ingeniería y Desarrollo*, nº 17, pp. 148-159, 2005.

[12] A. Alarcón-Aldana, J. S. González-Sanabria, and S. L. Rodriguez-Torres, "Guía para pymes desarrolladoras de software, basada en la norma ISO/IEC 15504," *Revista Virtual Universidad Católica del Norte*, nº 34, pp. 285-313, 2011.

[13] E. Avila-Domenech, A. Abad, and V. De la Cruz, "Delfdroid: metodología ágil de desarrollo de software para dispositivos móviles," *Revista Ingeniería UC*, vol. 20 (3), pp. 59-70, 2013.

[14] A. Qumer, and B. Henderson-Sellers, "Measuring agility and adoptability of agile methods: a 4-Dimensional Analytical Tool," in *IADIS International Conference Applied Computing*, San Sebastián, España, 2006.

[15] A. Qumer, and B. Henderson-Sellers, "Comparative evaluation of XP and SCRUM using the 4d analytical tool (4-DAT)," in *European and Mediterranean Conference on Information Systems (EMCIS)*, Costa Blanca, Alicante, España, 2006.