Article **ECORFAN Journal- Republic of Paraguay**
June 2018 Vol.4 No.6 29-35

Methodology for the UPFIM own software development

Metodología para Desarrollo de Software Propio de la UPFIM

REYNA-ANGELES, Omar†*, HERNÁNDEZ-TAPIA, Zaila, SOTO-FERNÁNDEZ, Susana Leticia and GÓMEZ-RAMOS, Marcos Yamir

Universidad Politenica de Francisco I Madero

ID 1st Author: *Omar, Reyna-Angeles* / **ORC ID**: 0000-0001-6604-9059, **Researcher ID Thomson**: I-3308-2018, **CVU CONACYT ID**: 097627

ID 1st Coauthor: *Zaila, Hernández-Tapia* / **ORC ID**: 0000-0003-2564-3451, **Researcher ID Thomson**: G-6592-2018, **CVU CONACYT ID**: 622127

ID 2nd Coauthor: *Susana Leticia, Soto-Fernández* / **ORC ID**: 0000-0001-9119-8750, **Researcher ID Thomson**: G-5810-2018, **CVU CONACYT ID**: 622134

ID 3rd Coauthor: *Marcos Yamir, Gómez-Ramos* / **ORC ID**: 0000-0003-0797-2534, **Researcher ID Thomson**: H-3374-2018, **CVU CONACYT ID**: 622036

Received March 22, 2018; Accepted June 30, 2018

Abstract

In the Computer System Engineering (CSE) of the Universidad Politécnica de Francisco I. Madero (UPFIM), it is necessary to define a methodology for its own software development that allows to manage the progress in an appropriate way, to have the control over the future maintenance and creation of different versions of it; Teachers of the CSE Full Time Teacher category (FTT) and some Professors by Subject (PS) work on software development projects useful for some areas and departments of the UPFIM, because there is not enough budget to hire external the development of software or acquire it, however the automation of several processes is necessary. The methodology created has 4 stages: conception, production, growth and delivery. Each one of them has defined activities that can be carried out in parallel, the other is necessary to work them linearly, it can be considered semi-incremental since some activities and / or stages can be worked like this.

Methodology for software Development, Agile methodology, Own software

Resumen

En la Ingeniería en Sistemas Computacionales (ISC) de la Universidad Politécnica de Francisco I. Madero (UPFIM) es necesario definir una metodología para el desarrollo de software propio que permita administrar los desarrollos de manera adecuada, tener un control sobre el futuro mantenimiento y creación de distintas versiones del mismo; los docentes de la ISC categoría Profesor Tiempo Completo (PTC) y algunos Profesores por Asignatura (PA) trabajan en proyectos de desarrollo de software útil para algunas áreas y departamentos de la UPFIM, esto debido a que no se cuenta con presupuesto suficiente para poder contratar de manera externa el desarrollo de software o adquirirlo, sin embargo la automatización de varios procesos es necesaria. La metodología creada cuenta con 4 etapas: concepción, producción, maduración y entrega. Cada una de ellas tienen definidas actividades que se pueden realizar en paralelo, otras es necesario trabajarlas linealmente, se puede considerar semi-incremental ya que algunas actividades y/o etapas se pueden trabajar así.

Metodologias de desarrollo, Metodología ágil, Software propio

Citation: REYNA-ANGELES, Omar, HERNÁNDEZ-TAPIA, Zaila, SOTO-FERNÁNDEZ, Susana Leticia and GÓMEZ-RAMOS, Marcos Yamir. Methodology for the UPFIM own software development. ECORFAN Journal-Republic of Paraguay 2018, 4-6: 29-35.

* Correspondence to Author (email: oreyna@upfim.edu.mx)
† Researcher contributing first author.

Introduction

UPFIM has the Subdirección de Sistemas Informáticos (SSI), which is in charge of the administration of ICTs in the institution, one of the responsibilities of this area is the development of software and the maintenance of computer systems, due to the fact that the area is very limited by the number of personnel it has, practically only they administer and maintain the integral system of academic control (SICA), one person is in charge of the system code and another of the database. However, the university has many needs for process automation that can help the different areas to provide a better service to the almost 2500 students enrolled in the 8 educational programs it has.

One of the strategies that has been thought is that the teachers of the ISC's educational program can participate in their own software development projects for the different areas, this is due to the lack of budgetary sufficiency to acquire applications that resolve the lack of automation or hire a company for development. Currently the area does not have a development methodology because, as mentioned before, it only maintains the SICA.

The foregoing makes it essential to have a methodology for the development of sofwtare projects that considers stages, activities, steps to be followed, responsibilities, documentation to generate and nomentlatura of documents.

Failure to do so would not have procedures to follow in the development of the systems and their documentation, if the developers have experience they could achieve functional applications but poorly documented, which would cause great uncertainty in the future, since at some point the applications will require maintenance or updating, that is, once the system is put into operation the modification of a functional requirement, a non-functional requirement or the implementation of new requirements, it will become a problem since it will not be possible to implement a progress control of versions, nor of specifications.

We propose a current and innovative methodology appropriate to the organizational and functional needs of two areas (SSI and ISC) that would now work together in the development of applications: the sub-direction of computer systems and the direction of the educational program of Computer Systems Engineering UPFIM.

The methodology will allow to standardize the development of systems and optimize development times since the processes will be well defined with the appropriate controls for monitoring and compliance with the activities of the development team.

The methodology will be based on ideas and foundations of other existing ones, specifically RUP and the main agile methodologies such as, Extreme Programming, Scrum, Crystal and DSDM. Having this methodology will allow the generation of a correct and complete technical documentation to achieve an appropriate and integrated software change control.

The article is organized as follows: Section 2 presents the state of the art. Section 3 contains the general elements of the proposed methodology. In section 4 each of the stages of the methodology are defined and finally in section 5 the results are established on time and the future work.

State of the Art

The revision of different current methodologies has led to the definition of the state of the art, first the RUP (Rational Unified Process) was revised, understanding it as a robust and traditional methodology, after which the four most used agile methodologies were analyzed and from which more information is available: XP (Extreme Programming), Scrum, Crystal Clear, and DSDM (Dynamic Systems Development Method).

This work at the end of accounts is about proposing a hybrid methodology based on RUP and lightweight methodologies also called agile. Therefore, the in-depth knowledge of the two approaches, their applications, characteristics and methods is important.

Table 1 shows the main differences of the agile methodologies with respect to the traditional ones. These differences affect not only the process itself, but also the context of the development team as well as its organization.

ISSN-On line: 2414-4827

Metodologías Tradicionales y RUP	Metodologías Ágiles
Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo	Basadas en heurísticas provenientes de prácticas de producción de código
Cierta resistencia a los cambios	Especialmente preparados para cambios durante el proyecto
Impuestas externamente	Impuestas internamente (por el equipo de desarrollo)
Proceso mucho más controlado, con numerosas políticas/normas	Proceso menos controlado, con pocos principios
Requiere extensa documentación para controlar todo el flujo del desarrollo	Poca documentación, el control es mínimo pues se basan en la disciplina y capacidad de los desarrolladores.
Documenta todo para tener un buen control de versiones y cambios una vez ya liberado el sistema.	No se generan documentos a menos que sean muy importantes, para esta filosofía la documentación no es esencial.
Existe un contrato prefijado	No existe contrato tradicional o al menos es bastante flexible
El cliente interactúa con el equipo de desarrollo mediante reuniones	El cliente es parte del equipo de desarrollo
Grupos grandes y posiblemente distribuidos	Grupos pequeños (<10 integrantes) y trabajando en el mismo sitio
Más artefactos	Pocos artefactos
Más roles	Pocos roles
La arquitectura del software es esencial y se expresa mediante modelos	Menos énfasis en la arquitectura del software

Table 1 Differences between traditional methodologies and RUP vs Agile Methodologies
Source: Self Made

According to the analysis made of the existing methodologies, no one can apply for the SSI of the UPFIM, since none is completely adapted to the needs, limitations, resources and structure of the organization.

On the one hand, traditional methodologies and RUPs are impossible to implement as they require relatively large development teams with too many well-defined roles, and this organizational structure does not exist in SSI, mainly because these methodologies are more focused on software development companies than they provide outsourcing service to different clients.

The disadvantage of agile methodologies to fully consider them is that they do not consider documentation as important, and this is the problem that is intended to be solved with the proposal.

Therefore, both philosophies should take the characteristics that are adaptable to the SSI and generate a current and innovative methodology that meets the needs.

General elements of the methodology

The proposed methodology that from now on we will call Agile Methodology in Own Software Development (MADSP) is of the incremental type in the middle of its stages since with that it allows a flexibility in when to the changes during the development, there are functional products of It is possible to work in a modular way if necessary, to finally integrate everything in a single system. It is considered hybrid because elements of traditional, RUP and agile methodologies are taken into account, Following are the characteristics that were taken from each philosophy:

Agile Type

a. Prepared for changes during the project
b. There is no traditional contract with the client, because it is not necessary
c. Few roles
d. Small groups working in the same place
e. Good communication between the development group
f. Great capacity and experience in the development group
g. The client is part of the development group and is in the workplace
h. Based on the production of functional applications as soon as possible, considering only the essential documentation for control of changes in future versions
i. Little control, since it is based on the discipline of the development group.
j. RUP
k. The artifacts to be used as inputs and outputs of the stages are some of those used in this methodology.
l. The controls that will be carried out although they are minimum are based on those proposed by RUP.

As for the working group, it is recommended that they be in the same physical space to achieve better communication and work in pairs as recommended by agile methodologies in small groups for development. The roles of leader, developer, client and consultant should be considered, all of them are part of the UPFIM, the leader and developers should be elements of the SSI or the ISC, the client is from the area requesting the application and the consultant is someone also from UPFIM that is aware of the regulations, regulations or laws that must be considered in the processes to be automated. The MADSP consists of four stages (conception, production, maturation and delivery) in figure 1 the diagram of the stages of the MADSP is shown.

ISSN-On line: 2414-4827

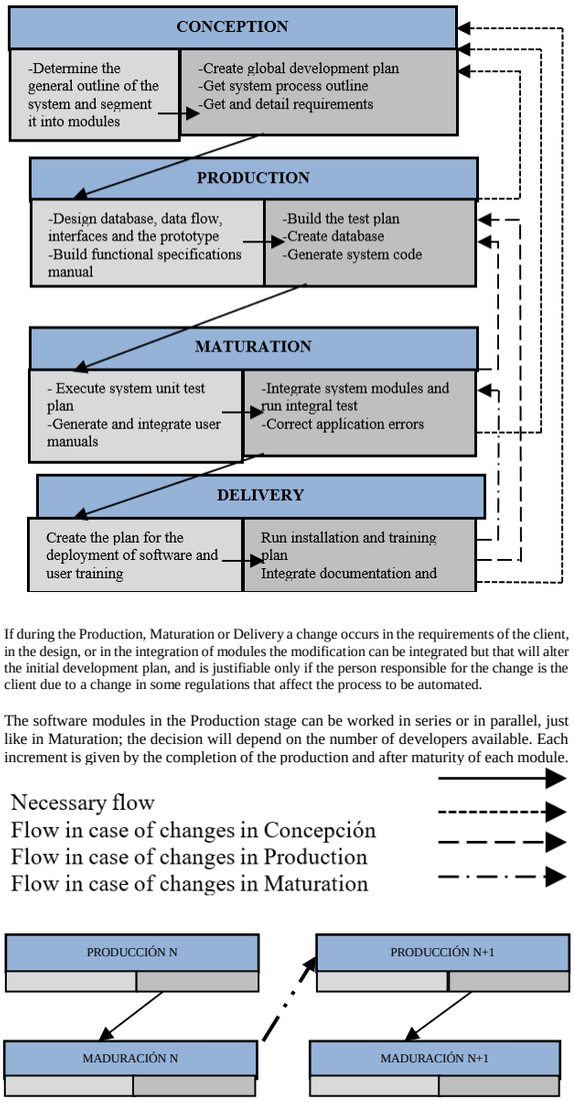

Figure 1 Diagram of the stages of MADSP
Source: Self Made

The Conception and Delivery do not work in increments, that is to say those stages are not finished until they are Completed, on the other hand the phases that if handled incrementally are Production and Maturation since they work by modules according to how many parts and small teams divided the development, this allows that at some point a team is working the production of a certain module and another team is in the maturation of another part of the software, or that all the equipment is in production or in maturation of different modules.

The methodology considers the possibility of changes in any of the stages, although this would affect the development plan and for that reason, as far as possible, they should be avoided, especially if the change is responsible for the change.

Figure 1 contains different types of lines that indicate the flow to follow in the process, only the continuous lines are the ones to follow during the development, the others will depend on a change in a previous stage or the number of developers available to participate in the creation of software Each stage is divided into two parts and each one of them indicates the work to be done. From one part to the other the workflow is serialized as indicated by the arrow, and within each of them the activities are worked in parallel.

Description of the stages of the MADSP

Conception

This first phase is not incremental, is made up of two parts like all the others, in the first is to determine the general scheme of the system and are segmented into modules this task is carried out by the Leader of the development team together with the client and the consultant's support if necessary.

After the above, the activities of the second part can be carried out, which are carried out in parallel, creating the overall development plan carried out by the group leader based on the general outline, the available personnel and the time in which the client requires the software, the leader and the developers are responsible for the task of obtaining the system's process schema. The task of compiling and detailing requirements is executed by the developers supported by the client (s) and the consultant. When detailing the requirements, those obtained by the leader in the general scheme of the system are used, the objective is not to leave any important details outside the requirements and be sure that all the functions to be implemented in the system have been captured.

Production

This stage is worked incrementally, it is considered as an increase the work with a module, that is to say the production of a module; For example, if the system will contain four modules, then there will be four iterations in the production one per module, being able to work more than one module in parallel as long as you have the number of developers available, remember that the developers must work in pairs , then to have at the same time two modules in Production, you must have 4 developers.

The activities of this stage are eight divided into two parts, the first consists of four activities and can be carried out in parallel.

ISSN-On line: 2414-4827

The developers are responsible for designing the system and here three activities are considered: the database, the data flow, the interfaces with the creation of a prototype; the fourth is carried out by the leader, build the specifications manual; the second part of the Production includes the other four activities that can be carried out in parallel, these are: build the test plan, develop database and generate complete code by system modules. In this stage is where you spend more time but having activities in parallel and work on modules helps streamline production.

The only activity where the client or the consultant participates here, is in the construction of the test plan as long as the leader considers it convenient.

The design of the database and the subsequent creation of it is done by a couple of developers only, while the design of the interface and the data flow, the construction of the test plan and the coding is divided into modules and into couples.

Maturation

Maturation is the third stage of the methodology and as the Production is worked incrementally and divided into two parts, with four activities being carried out in total, in the first part the unit testing plan is executed system and the user manuals are generated and integrated per module, in the second the integration of system modules is performed in parallel and run integral test, and the errors of the application that the test plan has thrown are corrected.

The execution of the test plan is carried out by modules, and it is carried out regularly by a expert expert in tests, here the client or the consultant can also participate if the leader considers it necessary at the suggestion of the expert in tests. The integration of the modules is done by a single team (couple) always having the necessary communication with the people who worked each module, in parallel the other pairs of developers generate the user manuals of the modules that corresponded to produce and mature. According to this methodology, you can have developers working on the production of a module, while others are already in the maturation of the module that corresponded to them. The integration of the modules will be given incrementally as the tests pass. The correction of errors will be made by the couple that was commissioned at the time to work on the module that has thrown the error.

Delivery

The delivery is the last stage and like the first one is not done incrementally, the activities are three, divided into two parts, in the first the leader creates the software installation plan and user training. In the second part in parallel, the installation and training plan of the system carried out by a couple of developers is executed, and another couple integrates the generated technical documentation and the release of the system is performed under a reception delivery with the client area.

Results and future work

So far we have worked on the tasks shown in Table 2, achieving the objectives of each activity.

Task	Objective
1. Carry out an analysis of the organization of the development team of UPFIM, to determine the requirements that the methodology to be proposed should cover.	Know the detail of the work area and obtain the profile of the human resources (developers of the SSI, developers of the ISC and the deputy director of computer systems)
2. Make an in-depth study of the software development methodologies Robust and Agile to determine the methods useful to the proposal that will be made.	Know and understand in depth the methodologies useful to the proposal that will be made. Determine that they will be taken from the known methodologies, to put together the proposal.
3.Define the generalities of methodology based on the principles, philosophies and phases of other existing methodologies, considering the context of the UPFIM development group and the needs of the SSI.	Generate the proposal of the MADSP to propose it to the SSI of the UPFIM in order to have a current and innovative methodology appropriate to their organizational and functional needs
4. Establish the stages of the methodology, the activities to be carried out in each of them, the work flow and communication between the activities of each stage and the sequence of the same.	

Table 2 Activities to define the development methodology of the SSI of UPFIM
Source: Self Made

Future work is defined in table 3 with the objective of each task

ISSN-On line: 2414-4827

Future task	Objective
1. Define for each stage description and format of the artifacts (records) of each activity, these can be printed or electronic and can refer to documents or application code, as well as the person in charge of generating and safeguarding them	Have elements of control during the development and document the different activities and stages for future system maintenance and version control
2. Establish a nomenclature for each device	Identify in a standardized way each record of the activities, stages and systems for a better control of the documentation and support to the maintenance and control of the versions of the different applications developed
3. Develop application for the administration of academic, scientific and technological production of UPFIM teaching staff	Apply the methodology in a software development case to evaluate its effectiveness and analyze results obtained

Table 3 Future works
Source: Self Made

References

Bernd Bruegge, Dutoit Allen H.(2002); Ingeniería de Software Orientado Objetos(1era ed.); México; Prentice Hall

Concepción Suárez Ramiro(2007); Metodología de Gestión de Proyectos en las Administraciones Públicas según ISO 10.006; Tesis; Universidad de Oviedo; (Disponible en: http://www.tdr.cesca.es/TDX/TDR_UOV/TESIS/AVAILABLE/TDR-0215108-122610//UOV0024TRCS.pdf Consultado: el 28 de Diciembre del 2017)

Diego Berrueta(2006); Programación Extrema y Software Libre; Artículo; (Disponible en: http://www.asturlinux.org/archivos/jornadas2006/ponencias/ProgExtrema_Berrueta/ponencia-sl-y-xp.pdf Consultado: el 23 de diciembre del 2017)

Gerardo Fernández Escribano(2002); Introducción a Extreme Programming; Artículo; (Disponible en; http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-XP.pdf Consultado: el 22 de diciembre del 2017)

Jacobson Ivar, Booch Grandy, Rumbaungh James(2004); El Proceso Unificado de Desarrollo de Software(1era ed.); México; Pearson Education

Julio César Rueda Chacón (2006); Aplicación de la Metodología Rup para el Desarrollo Rápido de Aplicaciones Basado en el Estándar J2EE; Tesis; Universidad de San Carlos de Guatemala; (Disponible en: ftp://ftp.itmerida.mx/.../Documentacion%20RUP/ Tesis%20Aplicacion%20RUP.pdf Consultado: el 2 de enero del 2018)

Martin Fowler(2003); The New Methodology; Artículo; (Disponible en; http://www.programacionextrema.org/articulos/newMethodology.es.html.Consultado: el 20 de diciembre del 2017)

Martínez Alejandro, Martínez Raúl (2006); Guía a Rational Unified Process; Escuela Politécnica Superior de Albacete; Artículo; (Disponible en: http://www.dsi.uclm.es/asignaturas/42551/trabajosAnteriores/Trabajo-Guia%20RUP.pdf Consultado: el 13 de diciembre del 2017)

Peláez Sánchez José Ignacio (2004), Introducción a la Ingeniería de Software; Universidad de Málaga; (Disponible en: http://www.lcc.uma.es/~jignacio/index_archivos/TEMA1.pdf. Consultado: el 7 de diciembre del 2017)

Peláez Sánchez José Ignacio (2004), Gestión de Proyectos de Software; Universidad de Málaga; (Disponible en: http://www.lcc.uma.es/~jignacio/index_archivos/TEMA2.pdf. Consultado: el 7 de diciembre del 2017)

Peláez Sánchez José Ignacio(2004), Comunicación en el Desarrollo de Sistemas; Universidad de Málaga; (Disponible en: http://www.lcc.uma.es/~jignacio/index_archivos/TEMA3.pdf. Consultado: el 9 de diciembre del 2017)

Peláez Sánchez José Ignacio (2004), Metodologías para el Desarrollo de Software; Universidad de Málaga; (Disponible en: http://www.lcc.uma.es/~jignacio/index_archivos/TEMA4.pdf. Consultado: el 12 de diciembre del 2017)

Pressman. Roger S.(2002); Ingeniería de Software(5ta ed.); México; Mc Graw Hill

Roxana Giandini, Gabriela Pérez, Claudia Pons (2009); Un lenguaje de Transformación específico para Modelos de Proceso del Negocio; Articulo; Universidad Abierta Interamericana de Buenos Aires; Universidad Nacional de La Plata; (Disponible en: http://imgbiblio.vaneduc.edu.ar/fulltext/files/TC104933.pdf; Consultado el 25 de Mayo del 2018)

ISSN-On line: 2414-4827

Schenone Marcelo Hernán; Diseño de una Metodología Ágil de Desarrollo de Software; Tesis; Universidad de Buenos Aires, Facultad de Ingeniería; 2004; (Disponible en: http://materias.fi.uba.ar/7500/schenone-tesisdegradoingenieriainformatica.pdf; Consultado: el 21 de diciembre del 2017)

Stephen A. White, PHD Derek Miers; BPMN Guía de Referencia y Modelado Edición Digital; 2009; (Disponible en: http://www.futstrat.com/books/BPMN_edicion_espanol.php; Consultado el 16 de mayo del 2018)

ISSN-On line: 2414-4827