METODOLOGÍA ÁGIL: EXTREME PROGRAMMING
AGILE METHODOLOGY: EXTREME PROGRAMMING

Melissa Álvarez Martell
melalvareez@gmail.com
Mariana Marcelino Aranda
mmarcelino@.mx
Magdalena Marciano Melchor
mmarciano@ipn.mx
Archibal Raziel Sánchez Peralta
asanchezp2008@alumno.ipn.mx

Instituto Politécnico Nacional-UPIICSA
Instituto Politécnico Nacional-CIDETEC
Instituto Politécnico Nacional-UPIICSA

Boletín No. 88
1o. de enero de 2022

Resumen

En la gestión de proyectos de software se observa una creciente tendencia en la adopción de metodologías ágiles. Las cuales buscan mitigar las desventajas que las metodologías tradicionales ocasionan en la industria. Al ser Extreme Programming (XP) uno de los mayores exponentes de las metodologías ágiles, se considera importante analizar qué características dentro de su planteamiento son las que lo posicionan como una metodología exitosa, lo anterior a partir de una revisión documental. XP se adapta a las necesidades del mundo actual, logra visualizar y resolver los impedimentos que afectan a las metodologías tradicionales mientras reta los riesgos propios de esta clase de proyectos, es decir, su completitud, volatilidad y complejidad. La combinación de metodologías da pie a una nueva categoría, los modelos híbridos, que prometen una mayor satisfacción de los usuarios y una mejora en los resultados.

Palabras Clave: Gestión de proyectos de software, Metodología ágil, Desarrollo de software.

Abstract

In software project management there is a growing trend in the adoption of agile methodologies. Which seek to mitigate the disadvantages that traditional methodologies cause on the industry. As Extreme Programming (XP) is one of the best agile methodologies' indicators, it's important to analyze what characteristics within its approach are those that position it as a successful methodology, the above based on a documentary review. XP adapts to today's world needs, it visualizes and resolve impediments that affect traditional methodologies while challenging its own risks in this

type of projects, that is, their completeness, volatility and complexity. The combination of certain methodologies brings new categories, hybrid models, which vow to give greater user satisfaction and improved results.

Keywords: Software project management, Agile methodology, Software development.

Introducción

La complejidad en la gestión de los procesos que involucra el desarrollo de proyectos de software implica para los equipos de trabajo, desde una alta posibilidad de retraso en entregas hasta un fracaso total del proyecto. En búsqueda de mitigar, en la medida de lo posible, los riesgos que conlleva un desarrollo software se generan metodologías que apoyan y dirigen los procesos del desarrollo mientras se magnifican los resultados.

Una metodología de gestión de proyectos es un conjunto de técnicas y métodos que abordan de forma detallada y completa cada una de las actividades del ciclo de vida de un proyecto de desarrollo de software. Es decir, funge como guía para obtener productos con un alto grado de calidad y satisfacción para ambas partes, cliente y equipo (Parada, 2016). Las metodologías de desarrollo software datan de los años cincuenta. Éstas se diferencian en dos grandes grupos, las metodologías tradicionales, que parten de los primeros preceptos establecidos para estas "guías" y las metodologías ágiles que se adaptan a las necesidades actuales de los proyectos de software mientras maximizan los procesos y resultados de las metodologías tradicionales.

Una metodología es "ágil" cuando se centra en las personas, se orienta a la comunicación, es flexible, rápida, eficiente, adaptable y aprende centrada en la mejora durante y después del desarrollo del proyecto de software (Ávila et al., 2013). "The Agile Alliance", desde el 2001, es la organización dedicada a promover los métodos ágiles en el desarrollo de software a través del "Manifiesto ágil". Algunos ejemplos son: Extreme Programming (XP), Scrum, Cristal Methodos (CM), Adaptive Software Development (ASD), Agile Modeling (AM), Agile RUP (dX), Dynamic Solutions Delivery Model (DSDM), Evolutionary Project Management (EVO), Feature Driven Development (FDD), Feature Driven Development (LD), entre otras (Parada, 2016; Mohammadi et al., 2008).

Al ser XP uno de los mayores exponentes de las metodologías ágiles, se considera importante analizar qué características dentro de su planteamiento son las que lo posicionan como un exitoso modelo y al mismo tiempo, como una metodología de referencia para futuras metodologías ágiles e híbridas emergentes.

Extreme Programming (XP)

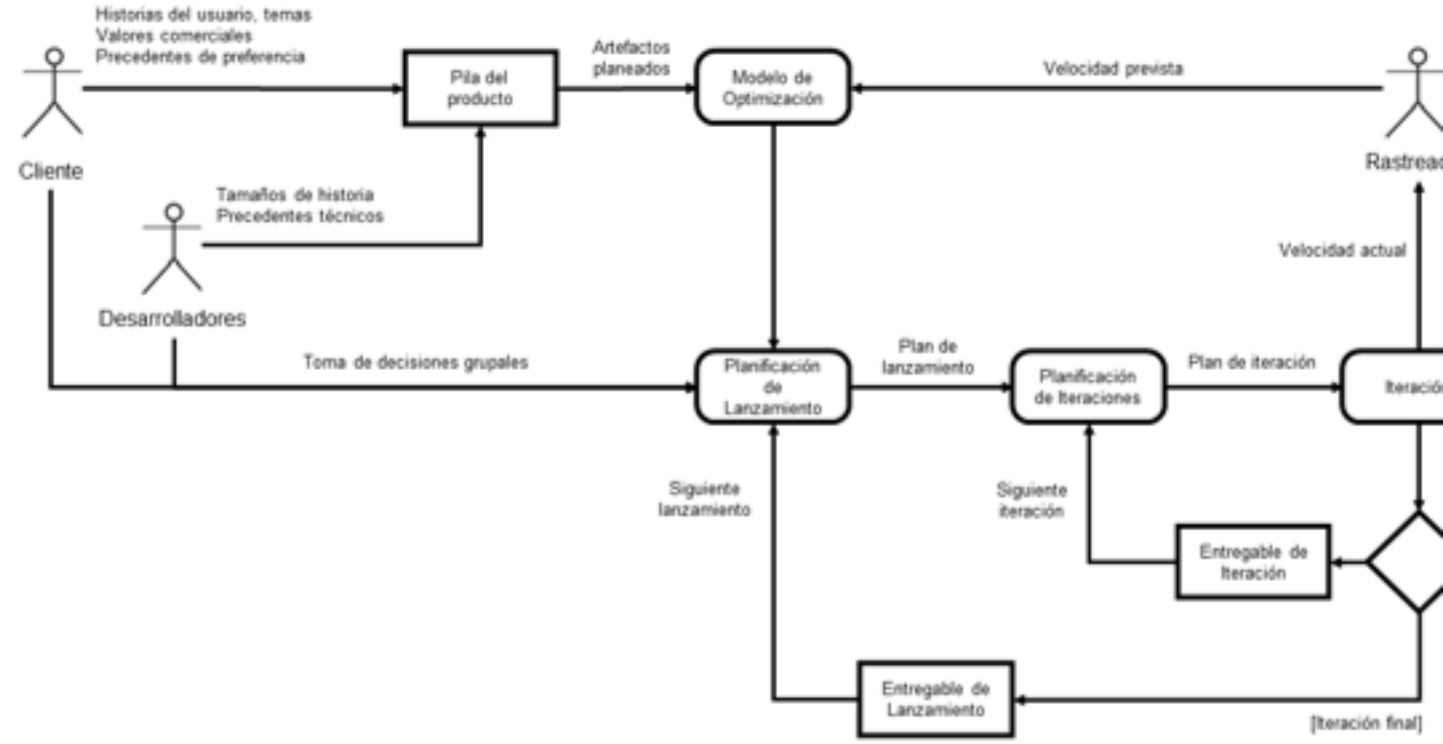
La programación extrema (XP) toma su nombre de su principio básico que es reducir al extremo el costo de los cambios (Bougroun et al., 2015; Blom, 2010), tuvo sus inicios en marzo de 1996 y surgió como una contrarreacción al enfoque creciente en los procesos, métodos y documentos en los cuales se basan las metodologías tradicionales (Kuz et al., 2018; Sharma et al., 2016; Siebra et al., 2008; Braithwaite et al., 2005). Es decir, XP es una colección de prácticas de ingeniería de software, donde el papel principal regresa al programador y el foco principal es el desarrollo de software.

Su propósito es entregar software de manera continua (software iterativo) mientras se mejora su calidad y responde a los requisitos cambiantes del cliente, incluso si tales requisitos surgen al final del ciclo de desarrollo. Su implementación tiene cuatro valores: 1) sencillez, 2) respeto, 3) coraje y 4) comunicación. Es decir, el desarrollo del proyecto está basado en principios de acciones de sentido común para llevar a cabo a buen término cada una de las actividades. Esto subyace a la necesidad de soporte de comunicación, colaboración e intercambio de conocimientos para compartir la experiencia del dominio entre el equipo y hacia el cliente (Bougroun et al., 2015; Crawford et al., 2008; Erickson et al., 2005). A través de esto, se mejora la productividad del proceso y del equipo.

El proceso XP tiene cuatro actividades básicas: 1) codificar, 2) probar, 3) escuchar y 4) depurar. En XP el cliente define el valor de negocio a implementar de acuerdo con sus necesidades y restricciones de tiempo y el programador estima el esfuerzo necesario para implementar el software. La participación activa del cliente en el desarrollo del proyecto, conlleva beneficios: 1) incrementa la calidad del producto, 2) aumenta la capacidad para negociar; 3) mejora la capacidad de resolver conflictos; 4) aumenta el desempeño del proyecto y aumenta la disposición a experimentar e improvisar en la búsqueda de

soluciones, 5) genera un menor riesgo de producir funcionalidades innecesarias o inaceptables, 6) se reduce la resistencia natural al cambio; y se 7) remedia la falta de capacidad de decisión en la gestión (Mohammadi et al., 2008).

Los valores y actividades básicas de XP conducen a 12 prácticas centrales, que permiten trabajar en conjunto y con una estructura de relación: 1) semana laboral de 40 horas, 2) estándares de codificación, 3) propiedad colectiva del código, 4) integración continua, 5) prueba continua, 6) cliente in situ, 7) programación en pareja, 8) juego de planificación, 9) refactorización, 10) diseño simple, 11) pequeños lanzamientos, 12) metáfora del sistema (ver figura 1).



(Van Valkenhoef et. al., 2011).

Componentes/ Artefactos XP

La metodología XP debe indicar los artefactos que se deben usar para el análisis y el diseño de la aplicación. Excepto por el código producido, en algunos casos (a consideración del cliente y desarrolladores) todos los otros artefactos que se generen durante el proceso de desarrollo se consideran como opcionales. Los artefactos XP son: 1) Historias de usuario, representan una breve descripción del comportamiento del sistema, se crea una por cada característica principal del sistema y se emplean para las estimaciones de tiempo y para el plan de lanzamiento; 2) "Task Card", las tarjetas de tareas representan los pasos reales de desarrollo necesarios para implementar una historia de usuario y responden a la pregunta de ¿cómo se debe hacer?; 3) Tarjetas CRC (Clase-Responsabilidad-Colaborador), una vez localizada una clase se define su nombre, sus responsabilidades (funcionalidad) y los colaboradores que trabajaran en ella (Parada, 2016).

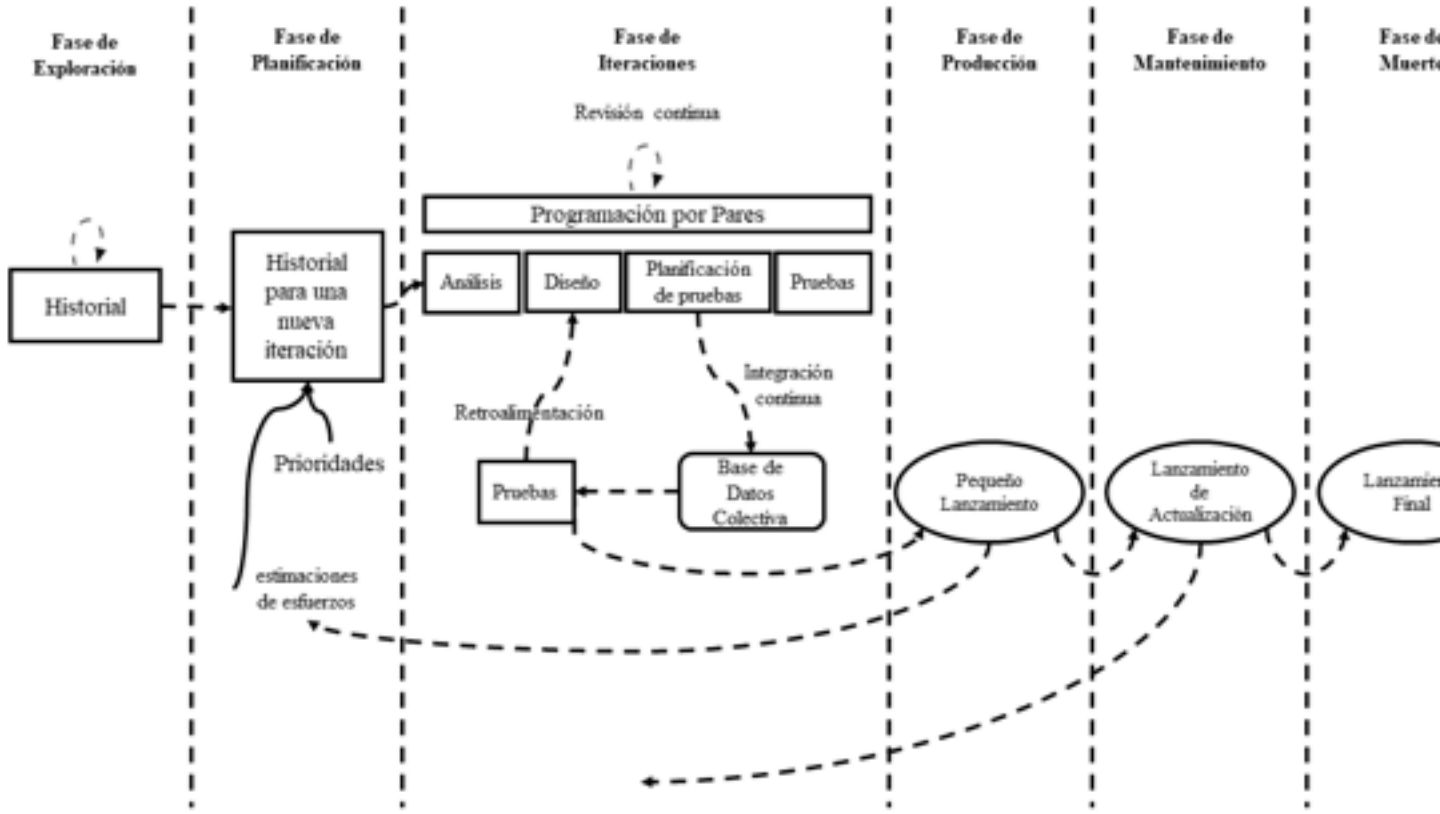
Roles

El proceso de desarrollo de software XP define siete roles: 1) Programador (Programmer), es el encargado de escribir el código fuente del sistema de software. 2) Cliente (Customer), escribe historias de usuario que sirven como guía al programador para realizar el código. 3) Encargado de pruebas (Tester), es el responsable de ayudar a los clientes a seleccionar y escribir pruebas funcionales. 4) Rastreador

(Tracker) realiza el seguimiento y estimaciones de todos los tiempos y entregas dentro del equipo. 5) Entrenador (Coach), persona responsable del proceso de desarrollo en su conjunto y se requiere que tenga experiencia en proyectos XP. 6) Consultor (Consultant), cuando el equipo necesita conocimientos especiales o específicos çontratan.ª un consultor que posea estos conocimientos y los transfiera a los miembros del equipo, para que ellos resuelvan el problema. 7) Gran jefe (Big boss), proporciona los recursos para llevar a cabo el proceso, tiene una visión general del proyecto, familiaridad con el estado del proyecto en todo momento y saber si cualquier intervención es necesaria para asegurar el éxito (Crawford et al., 2008).

Proceso XP e iteraciones

El proceso que rige a XP se basa en 6 fases: 1) Exploración, donde se analiza el historial, 2) Planificación, se definen las iteraciones, prioridades y estimaciones de esfuerzo, 3) Iteraciones, se lleva a cabo la programación en pares de las iteraciones previamente establecidas, aquí se lleva a cabo la integración continua y la prueba de cada cambio, 4) Producción, se producen los pequeños lanzamientos, 5) Mantenimiento, se producen los lanzamientos que actualizan al sistema, 6) Muerte, en este punto el sistema finalizará y estará listo para su implementación total (ver figura 2).



(Crawford et al., 2008)

XP sigue una estrategia de prueba automatizada y restringe la propagación del riesgo en las etapas posteriores a la que se desarrolla en ese momento. Así mismo continua la verificación de ocurrencia de riesgo después de cada fase del desarrollo y lo resuelve en esa misma etapa del ciclo de desarrollo (Sharma et al., 2016).

V. Conclusión

Extreme programming es una metodología ágil que acepta y fomenta los constantes cambios de un proyecto. Aborda esta idea con compromiso a través de entregas incrementales, y aboga siempre que estas sean en tiempo y forma. Esta metodología se adapta a las necesidades del mundo actual, logra visualizar y resolver los impedimentos que afectan a las metodologías tradicionales mientras reta los riesgos propios de esta clase de proyectos (completitud, volatilidad y complejidad).

La flexibilidad de esta metodología permite no solo facilitar los procesos de desarrollo de software, si no también innovar en sus procesos, y genera un debate respecto al perfeccionamiento de esta o la adaptación circunstancial de los diferentes proyectos en los que es implementada. Se adapta a proyectos de software de tamaño pequeño a mediano y, cuando es un proyecto de software a gran escala, el modelo en cascada puede tomarse como base (modelo híbrido).

La combinación de metodologías propicia a una nueva categoría, los modelos híbridos, que prometen una mayor satisfacción de los usuarios y una mejora en los resultados, al tomar las desventajas de los modelos tradicionales y ágiles, solucionarlos y potenciar los resultados de la aplicación. Lo cual, abre un panorama en la creación de nuevas metodologías para diversas aplicaciones, donde como en cualquier innovación, esta deberá ser implementada, comprobada, comparada y como hasta ahora, pueda ser mejorada.

Referencias

1. Ávila, E., & Meneses, A. (**2013**). *Delfdroid y su comparación evaluativa con XP y Scrum mediante el método 4-DAT. Revista Cubana de Ciencias Informáticas,* 7(1), 16–23. http://scielo.sld.cu/pdf/rcci/v7n1/rcci03113.pdf

2. Blom, M. (**2010**). *Is Scrum and XP suitable for CSE development? Procedia Computer Science,* 1(1), 1511–1517. https://doi.org/10.1016/j.procs.2010.04.168

3. Bougroun, Z., Zeaaraoui, A., & Bouchentouf, T. (**2015**). *The projection of the specific practices of the third level of CMMI model in agile methods: Scrum, XP and Kanban. Colloquium in Information Science and Technology, CIST,* (pp. 174–179). https://doi.org/10.1109/CIST.2014.7016614

4. Braithwaite, K., & Joyce, T. (**2015**). *XP expanded: Distributed extreme programming. Lecture Notes in Computer Science,* 3556, 180–188. https://doi.org/10.1007/11499053_21

5. Crawford, B., & Le, C. (**2008**). *Does eXtreme programming support collaborative creativity? Proceedings of the International MultiConference of Engineers and Computer Scientists, IMECS,* (pp. 19–21). http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.149.3694&rep=rep1&type=pdf

6. Erickson, J., Lyytinen, K., & Siau, K. (**2005**). *Agile modeling, agile software development, and extreme programming: the state of research. Journal of Database Management,* 16(4), 88-100. https://doi.org/10.4018/jdm.2005100105

7. Kuz, A., Falco, M., & Giandini, R. S. ((**2018**). *Comprendiendo la aplicabilidad de Scrum en el aula: Herramientas y ejemplos. Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología,* 21, e07. https://doi.org/10.24215/18509959.21.e07

8. Mohammadi, S., Nikkhahan, B., & Sohrabi, S. (**2008**). *). An analytical survey of "on-site customer" practice in extreme programming. International Symposium on Computer Science and its Applications, CSA,* (pp. 1–6). https://doi.org/10.1109/CSA.2008.72

9. Parada, C. (**2016**). *Caracterización de las metodologías ágiles para el desarrollo de aplicaciones móviles. Universidad Francisco de Paula Santander,* 1–6.

10. Sharma, P., & Hasteer, N. (**2016**). *Analysis of linear sequential and extreme programming development methodology for a gaming application. International Conference on Communication and Signal Processing, ICCSP,* (pp. 1916–1920). https://doi.org/10.1109/ICCSP.2016.7754505

11. Siebra, C. A., Filho, M. S. A., Silva, F. Q. B., & Santos, A. L. M. (**2008**). *Deciphering extreme programming practices for innovation process management. 4th IEEE International Conference on Management of Innovation and Technology, ICMIT,* (pp. 1292–1297). https://doi.org/10.1109/ICMIT.2008.4654557

12. Van Valkenhoef, G., Tervonen, T., De Brock, B., & Postmus, D. (**2011**). *Quantitative release planning in extreme programming. Information and Software Technology,* 53(11), 1227–1235. https://doi.org/10.1016/j.infsof.2011.05.007