Análisis de las características y funciones de las metodologías ágiles para el desarrollo de software que se implementan en un entorno empresarial

Narváez Díaz Jesús Francisco. Ruales Ortega Duban Bladimir

Estudiante de Desarrollo de Software

Estudiante de Desarrollo de Software

Jesusnarvaez2020@itp.edu.co, duvanruales2020@itp.edu.co

Resumen

El concepto de metodologías ágiles ha generado un gran interés en las organizaciones por lograr objetivos de desarrollo de software rápido y funcional, donde la importancia de implementar la metodología al equipo de trabajo o al entorno donde se desarrolle el proyecto, El rol fundamental de las metodologías es sin duda esencial en un proyecto, que debe encajar en el equipo, guiar y organizar actividades que conlleven a las metas trazadas en el grupo.

En el presente artículo se detallan los beneficios que las empresas adquieren al implementar las metodologías ágiles de desarrollo de software, tales como la metodología Extreme programming (XP),Iconix y Crystal las cuales ponen vital importancia en la capacidad de respuesta a los cambios, la confianza en las habilidades del equipo y al mantener una buena relación con el cliente, se conocerá cuáles son los alcances al implementar una metodología en los proyectos de software y cuales pueden integrarse al equipo de trabajo y los

proyectos de las empresas desarrolladoras de software, es por ello que, ofrecemos un estudio de las metodologías de desarrollo de software dejando libertad de elección para el lector el poder elegir la metodología que se adapte a su empresa de desarrollo.

Palabras clave: Metodologías ágiles, Desarrollo de software, Flexibilidad, Iterativo, Incremental.

Analysis of the characteristics and functions of agile methodologies for software development that are implemented in a business environment

Abstract

The concept of agile methodologies has generated a great interest in organizations to achieve objectives of rapid and functional software development, where the importance of implementing the methodology to the team or the environment where the project is developed, the fundamental role of methodologies is undoubtedly essential in a project, which must fit in the team, guide and organize activities that lead to the goals set in the group.

In this article we detail the benefits that companies acquire by implementing agile software development methodologies, such as Extreme programming (XP), Iconix and Crystal which put vital importance on the ability to respond to changes, confidence in the team's skills and maintaining a good relationship with the client, you will know which are

the scopes when implementing a methodology in the software projects and which can be integrated to the work team and the projects of the software development companies, that is why, we offer a study of the software development methodologies leaving freedom of choice for the reader to be able to choose the methodology that adapts to his development company.

Keywords: Agile methodologies, software development, flexibility, Iterative, Incremental.

Introducción

Los sistemas de software que son desarrollados en la actualidad son cada vez más grandes y de la misma manera más complejos debido a la gran demanda de los clientes que buscan adquirir software confiable, rápido y seguros, así mismo las mejoras de las nuevas tecnologías exigen que las metodologías ágiles adquieran una participación en la iteración de todos los equipos (Papadopoulos G, 2015). Dando paso a diseños de software complejos donde las partes que se interesan por el desarrollo del proyecto son lo suficientemente aptos para gestionarse fácilmente de manera que los sistemas de software no son entregados con demoras, y sin sobrepasar el presupuesto establecido.

El software que se llega a desarrollar sin una metodología ágil probablemente termina con los costos elevados, haciendo que la producción sea ineficaz, así mismo que la interacción

entre los miembros del equipo no sea eficiente. De la misma manera sin una metodología no controlamos el tiempo que se implementa en cada etapa, por ende, los proyectos se retrasan o puede ser que lo que terminan construyendo no cumple con los requerimientos. Cuando se desarrolla un software sin una metodología no se sabe en que están trabajando los miembros del equipo ni cuando termina cada actividad, no hay prioridades en lo que se hace, se podría estar haciendo varias cosas pequeñas que desde el punto de vista del producto final pueden ser innecesarias o de poco impacto.

Tener equipos auto organizados, entrelazados y una estructura organizativa plana permite que los equipos ágiles colaboren estrechamente sin complicaciones innecesarias (Papadopoulos G, 2015). Además, el hecho de implementar una metodología ágil nos ayuda a introducir cambios de la mejor manera, controlar los procesos y los beneficios, de igual forma la metodología nos brinda herramientas para controlar las etapas adecuadamente.

Métodos

Teniendo en cuenta los conocimientos de programación en estudiantes y personas con poca experiencia con respecto al conocimiento de las metodologías de desarrollo ideales para ser implementadas en un grupo de trabajo o en colaboración a la hora de desarrollar un proyecto, teniendo en cuenta lo anterior se ha optado por hacer uso de un método de investigación de escritorio, la cual implica el uso de información científica ya existente con fines de aumentar la eficacia de la investigación, primeramente se inicia por realizar búsquedas de artículos científicos de las metodologías ágiles en las cuales nos enfocamos

que son: Extreme Programming (XP), Iconix y Crystal de tal forma que permita el análisis y estudio de la información recolectada por medio de los buscadores académicos ScienceDirect y Pubmed teniendo en cuenta que se tomaron los más destacados de la búsqueda, de igual manera se obtuvieron deducciones que permitieron hacer una comparativa empleando lo que abarca cada una de ellas.

En el desarrollo de este análisis y tratando la información se deduce que metodologías de desarrollo se adaptan mejor en cada uno de sus campos de aplicación del entorno empresarial que lo requiera, para lograr un proyecto de calidad, con un funcionamiento óptimo favoreciendo al usuario y al equipo de desarrollo de la mejor manera.

Marco teórico

Los métodos ágiles son un subconjunto de métodos evolutivos y se basan en la mejora iterativa y los procesos de desarrollo oportunistas (Laurie M, 2010). Dentro de las metodologías ágiles se tiene presente una gran diversidad de metodologías, y cada metodología cuenta con características fundamentales para el desarrollo de proyectos, por consiguiente, las metodologías de desarrollo implementadas con frecuencia en el entorno laboral, ya sean empresas y desarrolladores independientes.

Las metodologías ágiles XP, Iconix y Crystal son implementadas con recurrencia en el entorno empresarial, debido a que generen cambios importantes en la gestión de proyectos, con el propósito de responder a la necesidad de los mercados actuales y dotar de garantías a las demandas principales (García M, *et al.* 2017).

METODOLOGIA EXTREME PROGRAMMING (XP)

Esta metodología ágil se adapta a los proyectos grandes y pequeños, transformándolos en procesos claros, entendibles y simples abarcando todos los aportes del equipo de desarrollo sin llevar consigo una mayor documentación del proyecto y siguiendo unas fases como guía a tener en cuenta al utilizar XP.

Fases de la metodología XP:

Planeación:

> La Metodología XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores. El proyecto comienza recopilando las historias de usuarios, las que constituyen a los tradicionales casos de uso. Una vez obtenidas estas historias de usuarios, los programadores evalúan rápidamente el tiempo de desarrollo de cada una.

Los conceptos básicos de la planificación son:

- Las Historias de Usuarios, las cuales son descritas por el cliente, en su propio lenguaje, como descripciones cortas de lo que el sistema debe realizar.
- El Plan de Entregas (Release Plan), establece que las historias de usuarios serán agrupadas para conformar una entrega y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto.

- Plan de Iteraciones (Iteración Plan), las historias de usuarios seleccionadas para cada entrega son desarrolladas y probadas en un ciclo de iteración, de acuerdo al orden preestablecido.
- Reuniones diarias de Seguimiento (Stand – Up Meeting), el objetivo es mantener la comunicación entre el equipo y compartir problemas y soluciones.

Diseño:

- La Metodología XP hace especial énfasis en los diseños simples y claros. Los conceptos más importantes de diseño en esta metodología son los siguientes:
- Simplicidad: Un diseño simple se implementa más rápidamente que uno complejo. Por ello XP propone implementar el diseño más simple posible que funcione.
- Soluciones "Spike": Cuando aparecen problemas técnicos, o cuando es difícil de estimar el tiempo para implementar una historia de usuario, pueden utilizarse pequeños programas de prueba (llamados "Spike"), para explorar diferentes soluciones.
- Recodificación ("Refactoring"): Consiste en escribir nuevamente parte del código de un programa, sin cambiar su funcionalidad, a los efectos de crearlo más simple, conciso y entendible. Las metodologías de XP sugieren re codificar cada vez que sea necesario.
- Metáforas: XP sugiere utilizar este concepto como una manera sencilla de explicar el propósito del proyecto, así como guiar la estructura del mismo. Una buena metáfora debe ser fácil de comprender para el cliente y a su vez debe tener suficiente contenido como para que sirva de guía a la arquitectura del proyecto.

Codificación:

- Disponibilidad del cliente, Uno de los requerimientos de XP es tener al cliente disponible durante todo el proyecto. No solamente como apoyo a los desarrolladores, sino formando parte del grupo. El Involucramiento del cliente es fundamental para que pueda desarrollarse un proyecto.
- Uso de estándares: XP promueve la programación basada en estándares, de manera que sea fácilmente entendible por todo el equipo, y que facilite la re codificación.
- Programación dirigida por las pruebas ("Test-Driven Programming"): primeramente se escribe los test que el sistema debe pasar. Luego, el desarrollo debe ser el mínimo necesario para pasar las pruebas previamente definidas. Las pruebas a los que se refiere esta práctica, son las pruebas unitarias, realizadas por los desarrolladores. La definición de estos test al comienzo, condiciona o "dirige" el desarrollo.
- La Programación en pares: al trabajar en pares se minimizan los errores y se logran mejores diseños, compensando la inversión en horas. El producto obtenido es por lo general de mejor calidad que cuando el desarrollo se realiza por programadores individuales.
- Integraciones permanentes: Todos los desarrolladores necesitan trabajar siempre con la "última versión". Realizar cambios o mejoras sobre versiones antiguas causan graves problemas, y retrasan al proyecto. Es por eso que XP promueve publicar lo antes posible las nuevas versiones, aunque no sean las últimas, siempre que estén libres de errores.
- Propiedad Colectiva del Código: En un proyecto XP, todo el equipo puede contribuir con nuevas ideas que apliquen a cualquier parte del proyecto. Asimismo, una pareja de

programadores puede cambiar el código que sea necesario para corregir problemas, agregar funciones o re codificar.

- Ritmo Sostenido: La Metodología XP indica que debe llevar un ritmo sostenido de trabajo. El concepto que se desea establecer con esta práctica es planificar el trabajo de forma a mantener un ritmo constante y razonable, sin sobrecargar al equipo.

Pruebas:

- Pruebas unitarias: Todos los módulos deben de pasar las pruebas unitarias antes de ser liberados o publicados. Por otra parte, como se mencionó anteriormente.
- Detección y Corrección de Errores: Cuando se encuentra un error ("Bug"), éste debe ser corregido inmediatamente y se deben tener precauciones para que errores similares no vuelvan a ocurrir. Asimismo, se vuelven hacer pruebas para verificar la corrección.
- Pruebas de Aceptación: Son creadas en base a las historias de usuarios, en cada ciclo de la iteración del desarrollo. El Cliente debe especificar uno o diversos escenarios para comprobar que una historia de usuario ha sido correctamente implementada.

METODOLOGÍA ÁGIL ICONIX

Iconix es un proceso simplificado en comparación con otros procesos más tradicionales, que unifica un conjunto de métodos de orientación a objetivo de abarcar todo el ciclo de vida de un proyecto.

Fases de la metodología Iconix:

Análisis de requisitos:

- Identificar en el mundo real los objetivos y todas las relaciones de agregación y generalización entre ellos. Utilizar un diagrama de clases de alto nivel definido como modelo de dominio.
- Presentar, en lo posible, una prototipación rápida de las interfaces del sistema, los diagramas de navegación, etc., de forma que los clientes puedan comprender mejor el sistema propuesto.

  Existen cuatro tipos de prototipos:

  - ✓ Prototipo de viabilidad: para probar la viabilidad de una tecnología específica aplicable a un sistema de información.
  - ✓ Prototipo de necesidades: utilizado para descubrir las necesidades de contenido de los usuarios con respecto a la empresa.
  - ✓ Prototipo de diseño: se usa para simular el diseño del sistema de información final centrándose en la forma y el funcionamiento del sistema deseado.
  - ✓ Prototipo de implementación: es una expresión de los prototipos de diseño donde el prototipo evoluciona directamente hacia el sistema de producción.
- Identificar los casos de uso del sistema mostrando los actores involucrados. Utilizar para representar el modelo de casos de uso.

Análisis y diseño preliminar

- Describir los casos de uso, como un flujo principal de acciones, pudiendo contener los flujos alternativos y los flujos de excepción. La clave principal a tener en cuenta al realizar esta actividad, es que no se debe perder mucho tiempo con la descripción

textual. Se debería usar un estilo consistente que sea adecuado al contexto del proyecto.

- Realizar un diagrama de robustez. Se debe ilustrar gráficamente las interacciones entre los objetos participantes de un caso de uso, este diagrama permite analizar el texto narrativo de cada caso de uso e identificar un conjunto inicial de objetos participantes de cada caso de uso.

El análisis de robustez ayuda a identificar los objetos que participaran en cada caso de uso. Estos objetos que forman parte de los diagramas de robustez se clasifican dentro de los tres tipos siguientes:

- ✓ Objetos de interfaz: usados por los actores para comunicarse con el sistema. Son con los que los actores interactúan con el sistema, generalmente como ventanas, pantalla, diálogos y menús.
- ✓ Objetos de la entidad: son objetos del modelo del dominio. Son a menudo tablas y archivos que contiene archivos para la ejecución de dicho caso de uso.
- ✓ Objetos de control: es la unión entre la interfaz y los objetos de la entidad. Sirven como conexión entre los usuarios y los datos. Los controles son "objetos reales" en un diseño, pero usualmente sirven como una especie de oficinista para asegurar que no se olvide ninguna funcionalidad del sistema la cual puede ser requerida por algún caso de uso.

Las reglas básicas que se deben aplicar al realizar los diagramas de análisis de robustez:

- ✓ Actores solo pueden comunicarse con objetos interfaz
- ✓ Las interfaces solo pueden comunicarse con controles y actores
- ✓ Los objetos de la entidad sólo pueden comunicarse con controles.
- ✓ Los controles se comunican con interfaces, objetos identidad y con otros controles pero nunca con actores.

- Actualizar el diagrama de clases ya definido en el modelo de dominio con las nuevas clases y atributos descubiertas en los diagramas de robustez.

El diseño:

- Especificar el comportamiento a través del diagrama de secuencia. Para cada caso de uso identificar los mensajes entre los diferentes objetos. Es necesario utilizar los diagramas de colaboración para representar la interacción entre los objetos.
- Terminar el modelo estático, adicionando los detalles del diseño en el diagrama de clases.
- Verificar si el diseño satisface los requisitos identificados.

La implementación:

- Utilizar el diagrama de componentes, si fuera necesario para apoyar el desarrollo. Es decir, mostrar la distribución física de los elementos que componen la estructura interna del sistema.
- Escribir y /o generar el código.

  También es necesario que se tenga en cuenta factores como:

  - ✓ La reusabilidad: que es la posibilidad de hacer uso de los componentes en diferentes aplicaciones.

- ✓ La extensibilidad: que consiste en modificar con facilidad el software.
- ✓ La confiabilidad: realización de sistemas descartando las posibilidades de error.

- Realizar pruebas. Test de unidades, de casos, datos y resultados. Test de interacción con los usuarios para verificar la aceptación de los resultados.

METODOLOGIA ÁGIL CRYSTAL

La metodología consiste en un conjunto de estrategias para el desarrollo donde se caracteriza por estar centrado en las personas que conforma el equipo de desarrollo y cliente teniendo presente la disminución al máximo del número de artefactos producidos. El equipo de desarrollo es un factor clave, por 10 que se deben invertir esfuerzos en mejorar sus habilidades y destrezas, así como tener políticas de trabajo bien definidos contemplando el tamaño del equipo, estableciéndose una clasificación por ejemplo Crystal Clear de 3 a 8 miembros y Crystal Orange de 25 a 50 miembros.

Los siete procesos o propiedades son:

- Entrega frecuente: Se hace prueba del desarrollo que se definieron en las etapas de entregas, los usuarios serán capaces de ofrecer información sobre los requisitos implementados, el cliente verá el progreso y los desarrolladores.
- Mejora continua: dar tiempo para que el equipo plantee dificultades y sobreponerse a los retos trazados para mejorar las cosas que no funcionan.

- Comunicación: Tener todo el equipo tan juntos (si es posible en el mismo puesto) con el fin de dar respuestas a las inquietudes y dificultades al instante.
- Seguridad Personal: La personas que conforman el equipo se sienta segura de hablar sin temor a represalias, que pueden dar críticas constructivas sobre el trabajo de otras personas y admitir sus propios errores, 10 que lleva a la honestidad y en última instancia a la confianza.
- Enfoque: Si todo el equipo tiene tiempo para centrarse en sus objetivos prioritarios dos horas al día, durante dos días consecutivos cada semana, sin ningún tipo de distracciones sin perder el norte (como reuniones o de otro trabajo), el equipo estará más enfocado y el trabajo.
- Fácil acceso a usuarios expertos: Si los usuarios expertos están disponibles para el equipo, pueden responder preguntas y ofrecer retroalimentación sobre la calidad y el diseño sobre las decisiones a tomar.
- Medio Ambiente, técnica de prueba automatizada, gestión e Integración frecuente: Un entorno técnico adecuado donde las tareas sean controladas por las pruebas y gestión de la configuración (versión - como hacer copias de seguridad y la fusión de los cambios).

Metodología	Características
Programación extrema (extreme	Metodología basada en prueba y error. Cliente bien definido.

programming, XP)	Los requisitos pueden y van a cambiar. Grupo pequeño y muy integrado (máximo 12 personas. Equipo con formación elevada y capacidad de aprender. Implementa compatibilidad y usabilidad con otras metodologías. Está orientada hacia quien produce y uso del software. Reduce el costo del cambio en todas las etapas del ciclo de vida del sistema. Combina las que han demostrado ser las mejores prácticas para desarrollar software y las lleva al extremo. Programación organizada. Reduce la taza de errores. Satisfacción del programador.
Crystal	Entregas frecuentes, en base a un ciclo de vida iterativo e incremental. Cuantas más personas estén implicadas, más grande debe ser la metodología. Entorno técnico con pruebas automatizadas, gestión de la configuración e integración continua. Si el proyecto tiene mucha densidad, un error no detectado puede ser crítico. El aumento de tamaño o densidad añade un coste considerable al proyecto. La forma más eficaz de comunicación es la interactiva (cara a cara). Tamaño de un equipo (número de componentes). Equipo Clear es para equipos de hasta 8 personas o menos. Amarillo para equipos entre 10 a 20 personas. Naranja para equipos entre 20 a 50 persona.
Iconix	Iterativo e incremental: varias iteraciones ocurren entre el desarrollo del modelo del dominio y la identificación de los casos de uso. Trazabilidad: cada paso está referenciado por algún requisito. Dinámica del UML: la metodología ofrece un “uso dinámico del UML” como los diagramas de caso de uso, etc.

Resultados y discusión

Los beneficios de implementar las prácticas de desarrollo de software es la capacidad de los equipos de desarrollo para adaptarse a los requisitos cambiantes de los clientes, a la vez que identifican y reducen ciertos riesgos que surgen durante el desarrollo de software (Rola et al, 2016). Las metodologías agiles se adaptan al entorno laboral donde los proyectos son medianos o grandes, asimismo la mescla adecuada entre arquitectura y desarrollo con agilidad representa una adaptabilidad a los procesos de desarrollo de software.

Las metodologías proporcionan flexibilidad y para muchos de los clientes representa una ventaja con la que pueden competir ante los demás y porqué estar preparados para el cambio significa que pueden reducir los costos. Las metodologías ágiles permiten a los desarrolladores retrasar las decisiones y una planificación adaptativa, por lo que el retrasar las decisiones tanto como sea posible de manera controlada será de cierta manera una ventaja tanto para el cliente como para la empresa, lo cual permitirá mantener al cliente satisfecho y así mismo el éxito del producto, por ese motivo las principales ventajas de poder retrasar las decisiones son.

- Se llega a reducir el número de decisiones de alta inversión que se toman.
- Reduce el número que se llega a implementar en el desarrollo del proyecto.
- Permite disminuir el costo de los cambios a implementar.

Además, las metodologías permite a los desarrolladores contar con entornos controlados, donde están preparados para los cambios que son introducidos en los procesos de desarrollo, así mismo hacer una planificación adaptativa consiste en tomar decisiones a lo largo del

proyecto, de esta manera se estará transformando el proyecto en un conjunto de proyectos pequeños, del mismo modo permite una planificación a corto plazo permite tener software disponible para los clientes y además ir haciendo la planificación más sensible, sea ante inconvenientes que aceleran o retrasan los productos.

Las metodologías agiles de desarrollo de software rescatan aspectos importantes al momento de integrarse en un entorno laboral como son: la reducción de al máximo de los artefactos producidos, toman la comunicación como aspecto indispensable, el equipo de desarrollo es un factor clave, las políticas dependerán del tamaño del equipo.

Como resultado de lo anterior se determinó de acuerdo a la información expuesta anteriormente que la implementación de estas metodologías se lleva a cabo teniendo en cuenta el equipo de desarrollo el cual es un factor clave, por lo que se debe identificar para que tipo de proyectos es apropiada una metodología ágil, en el caso de la metodología ágil Crystal que contiene una clasificación por colores, más específicamente, las cuales se adapta a campos de entre 1 a 8 miembros es determina como Clear Orange y de entre 25 a 50 miembros se determina como Clear Orange, entre otras.

En cuanto a la metodología ágil Xtreme Programming (XP) se tiene muy en cuenta al cliente, está enfocada en el diseño y principalmente en la programación al extremo es decir que con esta metodología se puede lograr terminar un proyecto en el menor tiempo posible al estipulado, basándose en proyectos pequeños y medianos también logra desarrollar excelente comunicación entre el equipo de desarrollo.

En lo referente a la metodología ágil Iconix la originalidad de esta metodología consiste en la definición de sus procesos, contrastando los requerimientos y la modelación del sistema, algo interesante es su flexibilidad para diferentes estilos y clases de problemas que se pueden presentar, se utiliza frecuentemente los casos de uso y también maneja una documentación considerable para llevar orden de la mayoría de los procesos que se elaboran.

Referencias bibliográficas

Papadopoulos, G. (2015). *Moving from traditional to agile software development methodologies also on large, distributed projects.* Procedia-Social and Behavioral Sciences. Obtenido de: https://www.sciencedirect.com/science/article/pii/S1877042815012835

Fojtik, R. (2011). *Extreme Programming in development of specific software. Procedia Computer Science. Odtenido de:* https://www.sciencedirect.com/science/article/pii/S1877050911000330

Papadopoulos, G. (2015). *Moving from Traditional to Agile Software Development Methodologies Also on Large, Distributed Projects.* Procedia - Social and Behavioral Sciences. Obtenido de: https://www.sciencedirect.com/science/article/pii/S1877042815012835

Losada, B., Urretavizcaya, M., & Castro, F. (2013). *A guide to agile development of interactive software with a "User Objectives"-driven methodology.* Science of Computer Programming. Obtenido de: https://www.sciencedirect.com/science/article/pii/S0167642312001657

Batra, D. (2020). *Job-work fit as a determinant of the acceptance of large-scale agile methodology.* Journal of Systems and Software. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S0164121220300583

Sousa, K., Vanderdonckt, J., Sellers, B., & Gonzales, C. (2012). *Evaluating a graphical notation for modelling software development methodologies.* Journal of Visual Languages & Computing. Obtenido de: https://www.sciencedirect.com/science/article/pii/S1045926X12000237

Patanakul, P., & Rufo, R. (2018). *Transitioning to agile software development: Lessons learned from a government-contracted program.* The Journal of High Technology Management Research. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S1047831018300294

Shameem, M., Kumar, R., Nadeem, M., & Khan, A. (2020). *Taxonomical classification of barriers for scaling agile methods in global software development environment using fuzzy analytic hierarchy process*. Applied Soft Computing. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S1568494620300624

Dingspyr, D., Nerur, S., Balijepally, V., & Moe, N. (2012). *A decade of agile methodologies: Towards explaining agile software development.* Journal of Systems and Software. Obtenido de: https://www.sciencedirect.com/science/article/pii/S0164121212000532

Taani, R., & Razali, R. (2013). *Prioritizing Requirements in Agile Development: A Conceptual Framework.* Procedia Technology. Obtenido de: https://www.sciencedirect.com/science/article/pii/S2212017313004064

Zaitsev, A., Gal, U., & Tan, B. (2020). *Coordination artifacts in Agile Software Development*. Information and Organization. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S1471772720300129

Galdwell, T. (2015). *Taking agile development beyond software – what are the security risks?*. Network Security. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S1353485815301100

Aldave, A., Vara, J., Granada, D., & Esperanza, M. (2019). *Leveraging creativity in requirements elicitation within agile software development: A systematic literature review.* Journal of Systems and Software. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S0164121219301712

Razavian, M., Paech, B., & Tang, A. (2019). *Empirical research for software architecture decision making: An analysis.* Journal of Systems and Software. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S016412121830267X

Laurie, W. (2010). *Agile Software Development Methodologies and Practices.* Advances in Computers. Obtenideo de: https://www.sciencedirect.com/science/article/pii/S0065245810800014

Kussunga, F & Ribeiro, P. (2019). *Proposal of a Visual Environment to Support Scrum. Procedia Computer Science. Obtenido de:* https://www.sciencedirect.com/science/article/pii/S1877050919322586

Longmuß, J & Höhne, B. (2017). Agile Learning for Vocationally Trained Expert Workers. Expanding Workplace-based Learning One Sprint at a Time. Procedia Manufacturing. Obtenido de: https://www.sciencedirect.com/science/article/pii/S235197891730121X

Blom, M. (2010). *Is Scrum and XP suitable for CSE Development?. Procedia Computer Science. Obtenido de:* *https://www.sciencedirect.com/science/article/pii/S1877050910001699*

Hidalgo, E. (2019). *Adapting the scrum framework for agile project management in science: case study of a distributed research initiative. Heliyon. Obtenido de:* *https://www.sciencedirect.com/science/article/pii/S2405844018340635*

Rola, P., Kuchta, D., & Kopczyk, D. (2016). *Conceptual model of working space for Agile (Scrum) project team. Journal of Systems and Software. Obtenido de:* *https://www.sciencedirect.com/science/article/abs/pii/S0164121216300401*

Chaouch, S., Mejri, A., & Ghannouchi, S. (2019). *A framework for risk management in Scrum development process. Procedia Computer Science. Obtenido de:* *https://www.sciencedirect.com/science/article/pii/S1877050919322100*

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). *The agile requirements refinery: Applying SCRUM principles to software product management. Information and Software Technology. Obtenido de:* *https://www.sciencedirect.com/science/article/abs/pii/S0950584910001539*

Howard, L., Farnaz, G., Pradeep, J., & Pinar, O. (2015). *A statistical analysis of the effects of Scrum and Kanban on software development projects. Robotics and Computer-Integrated Manufacturing. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S0736584515301599*

George, E. (2016). *Agile Project Management: Scrum, eXtreme Programming, and Scrumban. Project Management in Product Development. Obtenido de: https://www.sciencedirect.com/science/article/pii/B9780128023228000085*

Steghöfer, J., Burden, H., Alahyari, H., & Haneberg, D. (2017). *No Silver Brick: Opportunities and Limitations of Teaching Scrum with Lego Workshops. Journal of Systems and Software. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S0164121217301206*

Valkenhoef, G., Tervonen, T., Brock, B., & Postmus, D. (2011). *Quantitative release planning in extreme programming. Information and Software Technology. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S0950584911001340*

Tolfo, C., & Wazlawick, R. (2008). *The influence of organizational culture on the adoption of extreme programming. Journal of Systems and Software. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S0164121208000174*

Angioni, M., Carboni, D., Pinna, S., Sanna, R., Serra, N., & Soro, A. (2006). *Integrating XP project management in development environments. Journal of Systems Architecture. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S1383762106000646*

Dinesh, B. (2020). *Job-work fit as a determinant of the acceptance of large-scale agile methodology. Journal of Systems and Software. Obtenido de: https://www.sciencedirect.com/science/article/abs/pii/S0164121220300583*

Hernández, H., Ortega, E., & Lemus, C. (2014). Estimation and Control in Agile Methods for Software Development: a Case Study. Ingeniería, Investigación y Tecnología. Obtenido de: https://www.sciencedirect.com/science/article/pii/S1405774314703506