UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMATICA
Secretaria de Postgrado

GESTIÓN DE RIESGOS EN EL DESARROLLO ÁGIL DE SOFTWARE

Trabajo Final Integrador para obtener el grado de

"Especialista en Ingeniería de Software"

"Facultad de Informática - Universidad Nacional de La Plata"

Alumna: Lic. Andrea Peláez

Director: Lic. Alejandro Oliveros

Co-Directora: Mgter. Gladys Dapozo

Enero 2016

UNIVERSIDAD NACIONAL DE LA PLATA
FACULTAD DE INFORMÁTICA
Secretaría de Postgrado

"Si un proyecto no tiene riesgos, no lo hagas"

Tom Demarco & Timothy Lister

CONTENIDOS

LISTA DE FIGURAS

LISTA DE TABLAS

RESUMEN

Este trabajo explora los dominios *"Gestión de riesgos en proyectos de software"* y *"Metodologías ágiles de desarrollo"* dentro de la disciplina de Ingeniería de Software, y estudia los procesos ágiles de desarrollo bajo el enfoque de la gestión de riesgos, presentando los métodos y herramientas utilizados actualmente en los proyectos de software.

Los avances tecnológicos y las exigencias del mercado, dieron lugar a la aparición de nuevos métodos de gestión de procesos en los proyectos de desarrollo de software, debido a que los tradicionales no son suficientes dentro del escenario actual. Se han propuesto en los últimos años, modelos de procesos para hacer frente a la necesidad de una gestión más eficaz de riesgos en los proyectos de software.

La Ingeniería de software ha avanzado notablemente con la importación de prácticas y metodologías existentes en otras disciplinas, mejorando sus procesos y dando un viraje importante a la industria del software. El desarrollo ágil, evolucionó de las experiencias personales y el conocimiento colectivo de los consultores y referentes de la comunidad de software, y está transformando las formas en que se desarrolla software.

Las metodologías ágiles tienen una gran aceptación en la industria del software, no sólo se han incrementado los niveles de adopción, también se han integrado rápidamente a los principales enfoques de desarrollo tradicionales y de gestión de riesgos en proyectos de software.

La falta de consenso dentro de la comunidad ágil, ha llevado a muchos a creer que la gestión de riesgos es irrelevante en un modelo ágil, y que el progreso natural de su proceso iterativo es un medio suficiente para gestionar los riesgos. Lo cierto es que, el enfoque implícito de gestión de riesgos que ofrecen los modelos ágiles deja mucho margen de mejora.

Este trabajo avizora la necesidad de abordar una gestión de riesgos más explícita en los proyectos ágiles de desarrollo de software y aporta un panorama actual de las iniciativas y propuestas más prometedoras en ese sentido.

INTRODUCCIÓN

El crecimiento sostenido de la industria del software a lo largo de los últimos años, la posiciona como una de las más prometedoras para el funcionamiento de la economía nacional e internacional.

El desarrollo de software se organiza en el marco de un proyecto de ingeniería que involucra procesos técnicos y de gestión, en un esfuerzo destinado a cumplir un objetivo y plazo definido, y en el que intervienen otros factores: costos, riesgos, calidad y recursos.

La identificación a tiempo de incidentes, crisis o desviaciones, ha sido reconocida como uno de los diez factores críticos de éxito en la gestión de proyectos informáticos [1]. Según Pressman [2] *"Para administrar un proyecto de software exitoso, se debe comprender qué puede salir mal, de modo que los problemas puedan evitarse"*.

En una conferencia en 1968, se discutieron las experiencias de aplicar enfoques informales de desarrollo de software, que dejaron un saldo importante de: proyectos retrasados y/o superados en presupuestos, software de bajo desempeño e imposible de mantener, lo que se llamó la *"crisis del software"*. Desde entonces, se han realizado importantes aportes en prácticas, métodos y herramientas para abordar la complejidad inherente del software, y contribuir con la maduración de sus procesos de desarrollo.

En un estudio realizado con la participación de directores de proyectos de software para entender los factores que diferenciaban a los proyectos más exitosos de los menos exitosos, surge un patrón de observación: en los proyectos exitosos sus directores eran buenos gestores de riesgos [3]. No es casualidad que a finales de la década de los ochenta, el Project Management Institute (PMI) reconozca oficialmente a la gestión de riesgo de la dirección y gestión de proyectos como la parte más importante del cuerpo de conocimiento del Project Management (PMBOK) [4].

La selección de una estrategia de desarrollo inadecuada constituye una de las causas de fracaso de los proyectos, de allí la importancia de la elección inicial del modelo de desarrollo [5]. Las exigencias de un entorno de negocios cada vez más variable, complejo y global, llevan a adoptar nuevas experiencias y habilidades en el desarrollo de software. Los principios citados en el "Manifiesto Ágil[1]", promueven y conducen un conjunto de prácticas ágiles de desarrollo que van ganando aceptación en la comunidad del software.

[1] http://agilemanifesto.org

Las metodologías tradicionales tienden a acumular los riesgos que surgen en el desarrollo del producto al final del proyecto. Por otra parte, las prácticas ágiles priorizan los aspectos de desarrollo de alto riesgo al inicio del proyecto y las dificultades se reparten a lo largo del desarrollo en ciclos de corta duración [6]. La gestión de riesgos está implícita en los procesos ágiles y es realizada a través del progreso natural del proceso iterativo [7][8]. Cuando se selecciona la metodología de desarrollo para un proyecto software, se deben considerar entre otros aspectos, el riesgo del proyecto y el grado en que la metodología soporta la gestión del riesgo [9], pero esto no es suficiente para garantiza el éxito del proyecto, es necesario alcanzar un equilibrio entre los posibles riesgos y las potenciales oportunidades [10].

Esta investigación se propone establecer el estado del arte de la gestión de riesgos en el enfoque de las metodologías ágiles de desarrollo de software, y describir los métodos y herramientas utilizados actualmente en dicho dominio.

Este trabajo se organiza de la siguiente manera: en el *Primer Capítulo* se presenta un panorama actualizado de las áreas de investigación relacionadas con los riesgos en proyectos de software, principios y prácticas más utilizadas. En el *Segundo Capítulo* se describen los marcos de referencia y estándares de carácter genérico sobre gestión de riesgos, y los modelos de procesos de gestión de riesgos específicos de la ingeniería de software. En el *Tercer Capítulo* se analiza como la filosofía ágil está cambiando las formas en que se desarrolla software, y se describen los métodos ágiles más conocidos y utilizados en la industria del software. En el *Cuarto Capítulo* se exploran las fuentes de riesgo en los modelos de procesos más utilizados en el desarrollo de software y el grado en que cada modelo apoya la gestión de riesgos. En el *Quinto Capítulo*, se estudia la gestión de riesgos desde el enfoque particular de los procesos ágiles, se identifican y describen las experiencias reportadas en la utilización de métodos y herramientas para facilitar dicha gestión. Finalmente en el *Sexto Capítulo*, se exponen las *Conclusiones* y se proponen *Futuras líneas de investigación.*

OBJETIVOS DEL TRABAJO

Transformar el curso de las acciones de un proyecto de software en marcha, puede generar oportunidades de mejora ante acontecimientos inesperados que puedan presentarse. Todo cambio implica riesgo y tomar decisiones frente a él. Las metodologías ágiles de desarrollo son propuestas como una alternativa para responder a las altas tasas de cambio en requerimientos de software y expectativas de los clientes [11].

La gestión del riesgo es una de las zonas más tácitas en procesos ágiles [7], sin embargo no existe un consenso definitivo sobre la necesidad de la gestión de riesgos dentro de la comunidad ágil [8]. Este panorama motiva la revisión de las experiencias reportadas en artículos científicos de la especialidad y la construcción del estado del arte de la gestión de riesgos en proyectos de desarrollo ágil de software, a partir de los siguientes objetivos:

- Determinar los enfoques que ofrecen las metodologías ágiles para la gestión de riesgos en proyectos de desarrollo de software.
- Identificar, analizar y resumir los métodos y herramientas utilizados para la gestión de riesgos en el desarrollo ágil de software.

CAPÍTULO 1: GESTIÓN DE RIESGOS EN PROYECTOS DE SOFTWARE

Desaciertos en proyectos de software: Lecciones aprendidas

En la década del sesenta, la introducción de hardware más potente basado en circuitos integrados, hizo posible que los grandes y complejos desarrollos de software sean una propuesta factible. Sin embargo, los proyectos de software llegaban tarde, costaban más de lo presupuestado y con demasiados errores. *El desarrollo de software estaba en crisis, un enfoque informal de desarrollo no era suficiente, se necesitaban nuevas técnicas y métodos para controlar la complejidad inherente al software* [12].

El término *"crisis del software"* se mencionó por primera vez en el año 1968, cuando la organización OTAN (Organización de países del Tratado del Atlántico Norte) celebró la primera conferencia sobre desarrollo de software, en plena guerra fría y cuando el software constituía la base de las innovaciones militares de esa época. Se acuñó el término *"Ingeniería de Software" para describir el conjunto de conocimientos que existían en aquel estado inicial.* Desde entonces comienza a tener razón de ser el concepto de proyecto de software[2].

Brooks utilizó la expresión *"Bala de Plata"* en su artículo "No Silver Bullet — Essence and Accidents of *Software* Engineering" [13], para referirse a una solución perfecta al problema de la naturaleza del software, afirmando que en realidad no existe ninguna bala de plata, no hay ningún desarrollo, ni en tecnología ni en técnicas de gestión, que por sí solo prometa una mejora en un orden de magnitud en productividad, en fiabilidad, en simplicidad. Un *proyecto de software era capaz de convertirse en un monstruo de plazos, objetivos incumplidos, y productos defectuosos.* Brooks divide las dificultades o complejidades del desarrollo de software en *esenciales, dificultades inherentes a la naturaleza del software (complejidad, conformidad, variabilidad e invisibilidad)* y las dificultades *accidentales que se encuentran hoy en día pero que no son inherentes al software;* y sostiene que los avances en el pasado han reducido las dificultades *accidentales* de manera tal que el progreso ahora depende de ocuparse de las *esenciales.*

Ante el síndrome de la bala de plata, se buscan soluciones que de manera casi inmediata prometan acabar con el monstruo en el que en ocasiones se convierte un proyecto de software, tales como incrementar el personal asignado a un proyecto con la esperanza de

[2] http://www.scrummanager.net

disminuir el plazo de ejecución. En 1975 Frederick P. Brooks, en su libro "The Mythical Man Month", advierte que *"hombre-mes" como unidad para medir el tamaño de un trabajo, es un mito peligroso y engañoso*; e introduce una de las leyes más importantes en la administración de proyectos de software, conocida como la ley de Brooks: *"añadir recursos humanos a un proyecto retrasado lo hace demorarse aún más"*.

La ingeniería de software es una disciplina que comprende procesos técnicos de desarrollo de software, actividades tales como la gestión de proyectos de software y el desarrollo de herramientas, métodos y teorías de apoyo a la producción de software [12]. Las actividades del desarrollo de software se organizan en un proyecto, que tiene un inicio y un final definido, está restringido por la limitación de recursos y es ejecutado por personas. El resultado de un proyecto de desarrollo de software depende en gran medida de la forma en que se gestione [12]. "La aceptación de la dirección de proyectos como profesión indica que la aplicación de conocimientos, procesos, habilidades, herramientas y técnicas puede tener un impacto considerable en el éxito de un proyecto, el cual *debe medirse en términos de completar el proyecto dentro de las restricciones de alcance, tiempo, costo, calidad, recursos y riesgos*" [14].

La organización Standish Group[3] dedicada a la investigación y mejora del rendimiento de los proyectos de software, ha recopilado información en el entorno de las Tecnologías de Información (TI) y proyectos de desarrollo de software desde el año 1985. Con su reconocido Informe Chaos Report, se disparan innumerables debates en el ámbito de la ingeniería de software y sus estadísticas lo convierten en la referencia más citada en el sector. Su primer reporte publicado en 1994, muestra una tasa de éxito muy baja del 16%, 53% de los proyectos con deficiencias y 31% cancelados. Comparado con el reporte publicado en 2013 [15], se observa una evolución positiva hacia el 2012: 39 % corresponden a proyectos exitosos, 43% con deficiencias y 18% cancelados.

Si bien las estadísticas del Informe Chaos Report han tenido bastante repercusión en la comunidad del software, han suscitado polémicas discusiones en cuanto a considerar el éxito sólo cuando los proyectos se terminaron a tiempo, dentro del presupuesto y cumpliendo con los requerimientos, dejando fuera otros aspectos como la calidad, el riesgo y la satisfacción del cliente. El éxito del proyecto en TI ha mejorado cuando se examina en todos los ángulos que no están siendo considerados por el Informe Chaos. De todas formas, éste reporte sigue siendo una medida importante para la industria de TI [16].

[3] http://www.standishgroup.com

Riesgo en un proyecto de software

El software no es algo que se fabrica sino que se construye a la medida del usuario y se deteriora en el tiempo debido a los cambios [2]. El desarrollo de software es rico en oportunidades estratégicas, encierra una alta complejidad y está sujeto a altos niveles de incertidumbre [28].

El riesgo se define como *un evento o condición incierta, con consecuencias negativas en un proyecto de software* [31]. El riesgo en proyectos de software se determina en términos de su exposición a factores específicos, que presentan una amenaza para el logro de los resultados esperados de un proyecto. *El riesgo es el producto de la probabilidad del evento por el impacto o magnitud de la pérdida si se produce el evento* [19].

El 80% de los directivos asocian al riesgo con eventos negativos, encontrando en la capacidad de atraer opciones alternativas una única posibilidad de resultados positivos. *"El riesgo en sí mismo no es malo, riesgo es esencial para el progreso y el fracaso es a menudo una parte fundamental del aprendizaje"* [28]. El riesgo se ve como peligro o amenaza de un mal resultado, algunos investigadores definen el riesgo comprendiendo la amenaza y oportunidad [19]. En el mundo de los negocios el riesgo siempre está presente y se considera una condición para el progreso y la innovación; los riesgos surgen cuando se buscan oportunidades con limitados recursos en un contexto incierto y en continuo cambio [26].

El escenario de la complejidad en un proyecto de software, deriva indefectiblemente en incertidumbre de que el proyecto finalice dentro del presupuesto, tiempo y calidad prometida [29]. El nivel de incertidumbre es más elevado al inicio de un proyecto, la certeza de finalizarlo de forma satisfactoria aumenta gradualmente durante la vida del proyecto. *Un riesgo es una incertidumbre que puede tener un efecto negativo o positivo en el cumplimiento de los objetivos del proyecto* [14]. El riesgo es la posibilidad de que una acción elegida, incluida una inacción, de lugar a un resultado no deseado. *Los directores gestionan los proyectos en respuesta a la incertidumbre, tratando de lograr un equilibrio entre los riesgos y las oportunidades* [10].

Según Pressman [2], los *tipos de riesgos* que se pueden encontrar cuando se construye el software son:

Tabla 1: Tipos de Riesgos [2].

Tipo de Riesgo	Definición	Origen	Impacto
Riesgos del Proyecto	Amenazan el Plan del proyecto	Problemas de presupuesto, calendario, recursos, personal (técnico y de la organización).	Sobre el proyecto de software.

Tipo de Riesgo	Definición	Origen	Impacto
Riesgos Técnicos	Amenazan la calidad y temporalidad del software que se va a producir.	Problemas de diseño, implementación, interfaz, verificación, mantenimiento, ambigüedad en la especificación, incertidumbre técnica, obsolescencia técnica y la tecnología.	Sobre la implementación del producto software.
Riesgos Empresariales	Amenazan la viabilidad del software que se va a construir.	1) construir un producto o sistema excelente que realmente no se quiere (riesgo de mercado), 2) construir un producto que ya no encaje en la estrategia empresarial global de la compañía (riesgo estratégico), 3) construir un producto que el equipo de ventas no sabe cómo vender (riesgo de ventas), 4) perder el apoyo de los administradores debido a un cambio en el enfoque o en el personal (riesgo administrativo) 5) perder apoyo presupuestal o de personal (riesgos presupuestales).	Sobre el proyecto o el producto software.

Gerencia de Riesgos: minimizar las amenazas y maximizar las oportunidades

No hay temporada de riesgo o un equipo independiente para llevar adelante la gestión de riesgo, es un proceso que debe estar integrado en las actividades del proyecto de software [24]. Tom Demarco y Timothy Lister [25] afirman que nadie es inocente de mala praxis de gestión por no haber considerado el riesgo y comparan la gestión de riesgos en proyectos de software con un vals con osos, en una analogía con la disposición a tomar riesgos sin ceder oportunidades a los adversarios.

El riesgo afecta a los sucesos futuros, cambiando las acciones del presente se pueden generar oportunidades para conseguir una situación diferente en el mañana. El riesgo involucra una decisión y la incertidumbre que ella conlleva. *El análisis y la gestión del riesgo son acciones que ayudan al equipo de software a comprender y manejar la incertidumbre* [2]. Es importante gestionar la incertidumbre, es decir direccionar los riesgos o controlar sus efectos de una forma neutral o positiva dentro del proyecto [29].

La incertidumbre es la ausencia completa de información, sobre situaciones futuras que existen por fuera del control del gerente de proyecto y de su equipo. A mayor disponibilidad de información, mayor posibilidad de tomar una mejor decisión frente a los riesgos. El arte y la ciencia de la gerencia de riesgos consisten en analizar cómo resolver situaciones potenciales que pueden afectar significativamente el desempeño de un proyecto [30].

La gestión de riesgos es un proyecto dentro de un proyecto [28], que tiene como desafío minimizar las amenazas y maximizar las oportunidades. Una guía diseñada para ser utilizada por directores del departamento tecnológico y administradores de desarrollo de software, muestra como mediante una buena gerencia se pueden resolver los inconvenientes que aparecen dentro de las etapas de planeamiento, ejecución, implementación y entrega de los proyectos [18]:

Tabla 2: Obstáculos y alternativas de solución en la gestión de proyectos [18].

Etapas del proyecto	Obstáculos	Estrategias de solución
Planeamiento	Desconocimiento del negocio	• Grupo de trabajo conformado por personas que conozcan el funcionamiento de la empresa y los procesos a ser sistematizados. • El conocedor del negocio debe ser parte de los líderes del proyecto.
	Falta de claridad de los objetivos del proyecto	• Preparar a los líderes no sólo en el componente técnico sino también en temas administrativos. • Identificar cada grupo de trabajo y su participación en el resultado del proyecto, preparar reuniones con ellos, con el fin de explicar el objetivo y las herramientas para lograrlo. • Vender una visión de futuro deseado para la organización.
Ejecución	Resistencia al cambio y desestimación	• Asistencia del patrocinador a las reuniones críticas y demostración de respaldo al proyecto en público y en privado. • Asegurarse de que los directivos entienden los requerimientos para la realización del proyecto y que las personas están enfocadas en los resultados del negocio. • Identificar a las personas involucradas en todos los niveles de la organización y responsabilizarlas del éxito del proyecto.
Implementación	Condiciones del equipo de trabajo y sobrecarga	• Asegurarse de que el equipo de trabajo cuenta con el conocimiento necesario (técnico, operativo y administrativo).

Etapas del proyecto	Obstáculos	Estrategias de solución
		• Capacitar al grupo y contratar temporalmente personal experto cuando sea necesario. • Establecer prioridades y negociar con los directivos los nuevos requerimientos. • Realizar una valoración y crear más grupos de trabajo con nuevo personal en caso de ser necesario.
Entrega	La cultura organizacional frente a los cambios	• Identificar focos de resistencia y convencer de que no existen razones para estar confusos y asustados. • Empezar con pequeñas entregas en proyectos grandes evitando someter al nuevo usuario.

Los proyectos de desarrollo de software pueden ser generadores de cambio en la organización, la gestión de riesgos es importante para el logro de los objetivos del negocio, que está atado al éxito del proyecto. Los riesgos se presentan cuando las organizaciones persiguen oportunidades en un contexto de incertidumbre, el desafío es encontrar una posición apropiada ante los riesgos y aceptable para los interesados internos y externos en la búsqueda de los objetivos deseados. *La gestión de riesgos es una cuestión estratégica vinculada al compromiso, una estrategia de aversión al riesgo puede limitar el logro de los objetivos, por otra parte abrazar el riesgo puede aumentar las pérdidas del proyecto* [19].

Gestión eficaz del riesgo: principios y prácticas

"La gestión eficaz del riesgo es la herramienta más importante que un gestor de proyecto puede emplear para aumentar la probabilidad de éxito del proyecto" [24]. La implementación de procesos eficaces de gestión de riesgos tendrá éxito si cambia la cultura de la organización para motivar al individuo, dichos cambios requieren tiempo y la repetición de los procesos para que permanezcan firmemente arraigados en la organización. Los miembros del equipo pueden tener las habilidades necesarias para utilizar un proceso de gestión de riesgos, sin embargo esto no asegura que el equipo lo va a usar durante el ciclo de vida del proyecto; por otra parte un proceso documentado tampoco garantiza que se lo siga. La gestión eficaz del riesgo requiere de un comportamiento funcional, el cual se debe principalmente a la historia de la organización, la estructura, los procesos y el sistema de recompensa. *La alta dirección es la clave para el establecimiento de una organización que alienta el comportamiento de gestión de riesgos "funcional"* [24].

Las habilidades humanas se consideran a menudo suficientes para abordar los riesgos menos complejos. Al aumentar la complejidad del sistema, aumentan los riesgos y la necesidad de herramientas y métodos más sistemáticos para complementar el conocimiento, juicio y experiencia humana [28].

"La gestión de riesgos de proyectos de software se define como un conjunto de principios y prácticas encaminadas a la identificación, análisis y tratamiento de los factores de riesgo para mejorar las posibilidades de lograr un resultado exitoso del proyecto y/o evitar el fracaso del proyecto" [19]. A continuación se describen los *principios* y las *prácticas* más utilizadas actualmente en la gestión de riesgos en proyectos de software.

Principios aplicados a la gestión de riesgos

Solutions Framework de Microsoft (MSF) [74] reconoce a la administración de riesgos como una disciplina básica, que se basa en los siguientes principios fundamentales que contribuyen a la gestión efectiva de riesgos[4] del proyecto:

> *Mantenerse ágil, esperar el cambio:* La perspectiva de cambio es una de las principales fuentes de incertidumbre a la que se debe enfrentar el equipo de proyecto. Las actividades relacionadas con la administración de riesgos no deberían limitarse a una única fase del ciclo de vida de un proyecto. Con demasiada frecuencia, los equipos inician un proyecto con la intención de aplicar los principios básicos de la administración de riesgos, pero no logran sus objetivos debido a los ajustados tiempos de entrega. La agilidad exige que el equipo valore ininterrumpidamente y administre proactivamente los riesgos durante todas las fases del ciclo de vida de un proyecto porque los continuos cambios en todos los aspectos del proyecto significan que los riesgos también están cambiando. Un enfoque proactivo permite que el equipo acepte el cambio y lo convierta en una oportunidad, evitando así que se vuelva en su contra.
>
> *Fomentar la comunicación abierta:* MSF recomienda que los riesgos se discutan de forma abierta, tanto dentro del equipo como con los stakeholders claves externos al equipo. Todos los integrantes del equipo deben participar en la identificación y análisis de los riesgos. Los líderes del equipo y los responsables ejecutivos deben evitar que los riesgos se perciban como algo negativo y animar a los

[4] Los ocho principios son: 1. Mantenerse ágil y esperar el cambio. 2. Hacia una visión compartida del trabajo. 3. Entregar valor de negocio a lo largo del ciclo de vida. 4. invertir en calidad. 5. Aprender de todas las experiencias - buenas y malas, 6. Fomentar la comunicación abierta 7. Cuentas claras, responsabilidad compartida 8. Poder de equipo.

integrantes del equipo a que sigan este comportamiento. Los miembros de un equipo no deben tener reservas para comunicar sus opiniones con libertad para, de esta forma, evaluar con más precisión el estado del proyecto y tomar decisiones consensuadas entre los miembros del equipo y los patrocinadores.

Aprender de todas las experiencias: MSF da por hecho que el aprendizaje sólo puede ayudar a mejorar los resultados. El conocimiento obtenido en un proyecto puede reducir la incertidumbre de la toma de decisiones en otros proyectos cuando la información es poco fiable. MSF resalta la importancia que el aprendizaje de los resultados de otros proyectos tiene dentro de una organización. Por este motivo, en el proceso de administración de riesgos ha incorporado un paso relacionado con el aprendizaje. El análisis directo de los resultados de proyectos anteriores fomenta el aprendizaje dentro del equipo mediante el intercambio de opiniones entre los miembros del equipo.

Cuentas claras, responsabilidad compartida: No hay ninguna persona que sea "propietaria" de la administración de riesgos dentro de MSF. La participación activa en el proceso de administración de riesgos es responsabilidad de todos los integrantes del equipo. Los miembros del equipo tienen asignadas tareas específicas para analizar los riesgos en el marco del planeamiento general del proyecto, y cada uno de ellos se responsabiliza de llevarlas a cabo. Las actividades pueden afectar a todas las áreas del proyecto durante todas las fases de los ciclos del proyecto y del proceso de administración de riesgos. Estas actividades van desde la identificación de riesgos en áreas de experiencia o responsabilidad personal hasta el análisis de riesgos, el planeamiento de riesgos y la ejecución de tareas de control de riesgos durante el proyecto. En *MSF Team Model*, el área funcional de la administración del proyecto de la agrupación de funciones de administración del programa se responsabiliza de la organización del equipo en las actividades de administración de riesgos y que dichas actividades se incorporan en los procesos estándar de administración del proyecto.

Prácticas más utilizadas en gestión de riesgos

Entre los enfoques más utilizados en gestión de riesgo se mencionan las *listas de factores de riesgo de proyectos de software*, que constituyen una herramienta rápida y de bajo costo para la identificación y evaluación de la exposición de los riesgos del proyecto. Los factores de riesgo o de éxito más importantes en proyectos de software, generalmente se determinan a partir de encuestas de experiencias de los interesados. La percepción de dichos factores en los proyectos de software varía entre los grupos de interés, con el tiempo, a través de las etapas

del proyecto y del ciclo de vida, y entre las culturas. Se encuentran diferencias significativas en los factores identificados y su importancia relativa percibida en distintos entornos culturales [19].

Muchos de los factores de riesgos considerados importantes en la literatura, no son determinantes en las preocupaciones de toda la industria de software. Así surge de una investigación en la industria de software chilena. Del análisis de las respuestas obtenidas en una encuesta de percepción de proyectos y en una encuesta personal, dirigidas a jefes de proyectos y a distintos profesionales asociados al desarrollo, respectivamente; se determina que las personas consideran que el éxito de un proyecto de software no sólo está asociado al cumplimiento de un compromiso, como la satisfacción de un plan de proyecto, sino también a las condiciones de progreso laboral como del ambiente de trabajo de sus participantes una vez concluido el proyecto. Desde esta perspectiva, es necesario que la gestión de proyecto administre la motivación, las relaciones laborales y el cumplimiento de los compromisos [21].

Por lo general las *listas de control de factores de riesgo*, no son aplicables a todos los proyectos de software. Su mejor uso es como una lista de arranque en la evolución de un conjunto personalizado de factores de riesgo de los proyectos de software realizados en la organización a través del tiempo. Los factores de la lista genérica no considerados relevantes para los proyectos de la organización pueden ser reemplazados por factores que son identificados como riesgos. Las opiniones de todos los grupos de interés se deben tomar en cuenta en los procesos de identificación y revisión iterativa del riesgo en curso, no sólo durante la planificación del proyecto [19].

En lugar de trabajar simplemente a través de una lista predefinida de comprobación de factores específicos, un enfoque más amplio para pensar con un mayor nivel de abstracción y aprovechar el esfuerzo de gestionar los riesgos del proyecto de software, consiste en aplicar *los marcos analíticos que agrupan los factores de riesgos en categorías de acuerdo a los temas relacionados*. Las medidas de control de alcance individual, a menudo se pueden aplicar de manera efectiva a una o más categorías enteras de riesgo, en lugar de tratar cada factor individual. Las categorías pueden representar áreas objetivo para la aplicación de estrategias de control de riesgos [19].

En un esfuerzo por dar sentido a la cantidad de datos disponibles en las últimas décadas de continua investigación, algunos autores identificaron y clasificaron los factores que han demostrado tener impacto en los resultados del proyecto de desarrollo de software. McLeod y L. MacDonell (2011) formulan un marco analítico, que refleja el pensamiento contemporáneo de los factores que afectan los resultados de un proyecto de desarrollo de software. Para ello los agruparon en dimensiones conceptualmente manejables: contenido, proceso y contexto

del sistema relacionado con el cambio organizacional, y una cuarta dimensión basada en las personas y la acción, justificada en el creciente reconocimiento del desarrollo de software como un proceso social. La base de los diversos componentes del marco de clasificación, resultante de la revisión de la literatura entre 1996 y 2006, se complementa con una discusión sobre estudios empíricos, de la cual emergen la persistencia de ciertos factores tradicionales que influyen en el desarrollo de software, la naturaleza cambiante del desarrollo, la importancia relativa de las personas y los procesos en los resultados del proyecto, el reconocimiento de la importancia del contexto institucional en el que el desarrollo de software se lleva a cabo , y la necesidad de centrarse en las relaciones e interacciones entre los factores que influyen en los proyectos de software [22].

La gestión eficaz del riesgo requiere de administradores de proyectos que comprendan la naturaleza de los riesgos de software. La información sobre la probabilidad de ocurrencia e impacto de los riesgos de software en el desempeño del proyecto, puede ayudar al gerente a desarrollar una mejor estrategia de manejo del riesgo, a través de la evaluación de los niveles de riesgo, aplicando una *matriz de riesgo* en el proceso de análisis de riesgo del proyecto de software [31].

La *identificación de riesgos basado en Taxonomía* [32], es un método desarrollado a partir de la literatura publicada y la experiencia previa en el desarrollo de software y probado en proyectos de software civil y de defensa financiados por el Gobierno. El método consiste en un cuestionario llamado TBQ (Taxonomy-Based Questionnaire), y un procedimiento para su aplicación. El cuestionario asegura de que todas las áreas de riesgo se abordan sistemáticamente, mientras que el proceso de aplicación está diseñado para garantizar que las preguntas se hacen a las personas adecuadas y de la forma correcta para producir resultados óptimos.

Para las pequeñas organizaciones de desarrollo de software, se propone una herramienta de software para administrar la gestión de riesgos en proyectos de desarrollo de sistemas informáticos [27]. La herramienta web de uso libre, brinda la posibilidad de efectuar tareas de identificación de riesgos, planes de riesgos en base a taxonomías estándar, y planes de contingencia asociados a cada riesgo. La herramienta propuesta, fue desarrollada utilizando software de uso libre, permite analizar los riesgos Post Mortem al proyecto y documentar todas las desviaciones producidas, con la finalidad de crear una base de conocimiento que enriquezca futuros trabajos de la organización.

Las herramientas actuales de evaluación y gestión de riesgos para los proyectos de software son específicas del modelo de desarrollo utilizado. A partir de una revisión de la literatura, se detecta la necesidad *de herramientas inteligentes de gestión de riesgos* [28] que se puedan

utilizar con cualquier modelo de desarrollo de software: ágil, tradicional, o la combinación de ambos. En este sentido, se propone dos enfoques: *redes neuronales y agentes inteligentes, para el desarrollo de un modelo de herramienta de evaluación y gestión de riesgos*. Las redes neuronales como modelos de datos estadísticos o herramientas para la toma de decisiones, facilitan la identificación de las tendencias y patrones predictivos, y ponen de relieve las anomalías que pueden ser indicativas de riesgo. El uso de agentes inteligentes como una herramienta de gestión de riesgos, fomenta la aplicación de la integración de los métodos formales y enfoques heurísticos para garantizar el apoyo a la evaluación, comparación, análisis y evolución del comportamiento de los agentes.

CAPÍTULO 2: MODELOS DE GESTIÓN DE RIESGOS

Principales Modelos de Gestión de Riesgos

Los modelos de proceso especifican paso a paso las tareas para la gestión de riesgos. Conceptualmente, la mayoría de los modelos incluyen un conjunto similar de pasos de proceso: estrategia, identificación, análisis, respuesta y control de riesgos. Las listas de verificación, los marcos analíticos y los modelos de procesos están interrelacionados y se utilizan juntos a menudo. *Las listas de verificación y los marcos analíticos se pueden utilizar en los pasos de identificación y análisis de riesgos de un modelo de proceso. La principal contribución de los modelos de procesos es que guían la acción de gestión de riesgos en lugar de pensar sólo de forma analítica* [19].

En las Tablas 3 y 4, se identifican algunos de los marcos de referencia y estándares de carácter genérico sobre la gestión de riesgos, y los modelos de gestión de riesgos específicos de la ingeniería de software, respectivamente.

Tabla 3: Marcos de referencia y estándares genéricos de gestión de riesgos.

Marcos de referencia y estándares	Organización	Alcance	Enfoque de aplicación
Project Management BOdy of Knowledge (PMBOK)	Project Management Institute (PMI)	Gestión de proyectos (en diversos tipos de industrias).	El área de conocimiento "Gestión de riesgos del proyecto" incluye seis procesos relacionados con el desarrollo de la gestión de riesgos de un proyecto.
Risk IT	Information Systems Audit & Control Association (ISACA)	Gobierno de TI	Marco de referencia para la gestión de riesgos basado en principios, y compuesto de nueve procesos agrupados en tres dominios.
ISO 31000:2009 Risk Management	International Organization Standardization (ISO)	Organización pública o privada, comunidad, asociación, grupo o individuo.	Consta de tres elementos relacionados entre sí: principios de gestión de riesgos, el marco de gestión de riesgos y el proceso de gestión de riesgos.

Tabla 4: Modelos de gestión de riesgos específicos de la ingeniería de software.

Modelos de gestión de riesgos	Organización	Alcance	Enfoque de aplicación
Software Risk Management	Software Engineering	Proyecto de desarrollo de software	Se basa en principios y seis actividades que se desarrollan circularmente en

Modelos de gestión de riesgos	Organización	Alcance	Enfoque de aplicación
(SRM)	Institute (SEI)		un proceso continuo, donde la comunicación ocupa el conducto central por donde fluye toda la información.
Team Risk Management (TRM)	Software Engineering Institute (SEI)	Proyecto de desarrollo de software	Extiende el paradigma de gestión de riesgos SEI, incorporando cinco pasos de dicho paradigma en cuatro procesos. Se fundamenta en principios y se implementa como parte de la rutina, continua e integral de las actividades de gestión de proyectos.
CMMI (Capability Maturity Model Integration)	Software Engineering Institute (SEI)	Proyecto de desarrollo de software y servicios	La Gestión de Riesgos (RSKM) es un área de proceso de gestión de proyectos (Nivel de Madurez 3), se divide en tres metas, cada una de las cuales incluye prácticas específicas para alcanzarlas.
IEEE 1540:2001	Institute of Electrical and Electronics Engineers	Proyecto de desarrollo de software	Define un proceso de gestión de riesgos de software que se compone de varias actividades que funcionan de manera iterativa y que deben integrarse con otras prácticas y sistemas de gestión de riesgos de la organización
ISO/IEC 16085:2006	International Organization Standardization (ISO), International Electrotechnical Commission (IEC)	Proyecto de desarrollo de software	Sucesor del estándar IEEE 1540, propone un proceso continuo de gestión de riesgos y aborda sistemáticamente el riesgo en todo el ciclo de vida del software.

A continuación, se describe brevemente cada uno de los modelos mencionados en las tablas anteriores.

Marcos de referencia y Estándares de carácter genérico

Dentro de los marcos y las normas generales que se aplican para la gestión de riesgos en cualquier industria y sector, se pueden mencionar: *Project Management BOdy of Knowledge (PMBOK)* propuesto por el Project Management Institute (PMI), *Risk IT Framework* impulsado por ISACA y la norma *ISO 31000:2009* desarrollado por International Organization Standardization (ISO). Dichos marcos y normas se presentan brevemente a continuación.

PMBOK de Project Management Institute (PMI)

Los proyectos crean valor en forma de procesos de negocio mejorados, son indispensables para el desarrollo de nuevos productos y servicios, y facilitan a las organizaciones la respuesta ante los cambios del entorno, la competencia y el mercado. Los proyectos son imprescindibles para el crecimiento y supervivencia de las organizaciones, en este sentido la dirección de proyectos se convierte en una disciplina estratégica y la guía del PMBOK identifica los procesos para dirigir proyectos, en distintos tipos de industrias [14].

PMBOK es un estándar basado en prácticas contrastadas y de probado valor a nivel global, que evolucionó a partir de las buenas prácticas, reconocidas de los profesionales de todo el mundo que han contribuido a su desarrollo. PMBOK en su quinta edición, sigue reflejando la actualización de los conocimientos dentro de la profesión de Gestión de Proyectos, con el incremento del número de procesos a 47 y la creación de una nueva área de conocimiento llamada "*Project Stakeholder Management*" que refuerza su compromiso con los interesados del proyecto.

La complejidad creciente de los proyectos exige mayores aptitudes, conocimientos, y demostración de compromiso en el desempeño ético y profesional. PMI emite una certificación a nivel profesional para el rol de la gestión de riesgos, *"PMI Risk Management Professional (PMI-RMP)"* en reconocimiento a la disciplina y formación profesional.

"Los objetivos de la gestión de los riesgos del proyecto consisten en aumentar la probabilidad y el impacto de los eventos positivos, y disminuir la probabilidad y el impacto de los eventos negativos en el proyecto" [14].

La gestión de los riesgos del proyecto según el PMBOK, incluye los siguientes *procesos* [14]:

> *Planificar la Gestión de Riesgos:* Es el proceso de definir el plan de gestión de riesgos, para facilitar la comunicación y obtener el respaldo de todos los interesados a fin de asegurar que dicho proceso sea llevado a cabo de forma eficaz a lo largo del ciclo de vida del proyecto. La planificación proporciona los recursos y tiempo suficientes para realizar las actividades de gestión de riesgos, asegurando que el nivel, el tipo y la visibilidad de las mismas son acordes tanto con los riesgos como con la importancia del proyecto para la organización.
>
> *Identificar los Riesgos:* Es un proceso para determinar los riesgos que pueden afectar al proyecto y documentar sus características claramente y sin ambigüedades, con el objeto de llevar a cabo un análisis y un desarrollo de respuestas eficaces. Es un

proceso iterativo, los riesgos pueden evolucionar o descubrirse nuevos riesgos a medida que el proyecto avanza a lo largo de su ciclo de vida. Este proceso debe involucrar al equipo del proyecto para desarrollar un sentido de propiedad por los riesgos, el conocimiento y la capacidad para anticipar eventos, y el compromiso por las acciones de respuesta asociadas.

Realizar el Análisis Cualitativo de Riesgos: Es un proceso que permite a los directores de proyecto reducir el nivel de incertidumbre y concentrarse en riesgos de alta prioridad. Evalúa la prioridad de los riesgos identificados a través de la probabilidad relativa de ocurrencia, del impacto sobre los objetivos del proyecto si los riesgos llegaran a presentarse, así como otros factores, tales como el plazo de respuesta y la tolerancia al riesgo por parte de la organización, asociados con las restricciones del proyecto en términos de costo, cronograma, alcance y calidad.

Realizar el Análisis Cuantitativo de Riesgos: Es el proceso de analizar numéricamente el efecto de los riesgos identificados sobre los objetivos generales del proyecto, generando información cuantitativa sobre los riesgos para apoyar la toma de decisiones a fin de reducir la incertidumbre del proyecto. Esto es posible, en los casos en que se cuenta con datos suficientes para desarrollar los modelos adecuados, el director del proyecto debe utilizar el juicio de expertos para determinar la necesidad y viabilidad del análisis cuantitativo de riesgos. La disponibilidad de tiempo y presupuesto, así como la necesidad de declaraciones cualitativas o cuantitativas acerca de los riesgos y sus impactos, determinarán que método o métodos emplear para un proyecto específico. Se utiliza fundamentalmente para evaluar el efecto acumulativo de todos los riesgos que afectan al proyecto.

Planificar la Respuesta a los Riesgos: Es el proceso de desarrollar opciones y acciones para mejorar las oportunidades y reducir las amenazas a los objetivos del proyecto. Aborda los riesgos en función de su prioridad, introduciendo recursos y actividades en el presupuesto, el cronograma y el plan para la dirección del proyecto, según las necesidades. Cada respuesta a un riesgo requiere una comprensión del mecanismo por el cual se abordará el riesgo, dicho mecanismo es utilizado para analizar si el plan de respuesta a los riesgos está teniendo el efecto deseado, identificar y asignar una persona responsable de cada una de las respuestas a los riesgos acordadas y financiadas. Las respuestas a los riesgos deben adecuarse a la importancia del riesgo, ser rentables con relación al desafío a cumplir, realistas dentro del contexto del proyecto, acordadas por todas las partes involucradas y deben estar a cargo de una persona responsable.

Monitorear y controlar los Riesgos: Es el proceso de implementar los planes de respuesta a los riesgos, monitorear los riesgos identificados y los riesgos residuales, identificar riesgos nuevos, riesgos que cambian o que se tornan obsoletos, y evaluar la efectividad del proceso de gestión de riesgos a través del proyecto. Esto permite mejorar la eficiencia del enfoque de la gestión de riesgos a lo largo del ciclo de vida del proyecto, para optimizar de manera continua las respuestas a los riesgos, puede implicar la selección de estrategias alternativas, la ejecución de un plan de contingencia o de reserva, la implementación de acciones correctivas y la modificación del plan para la dirección del proyecto. El responsable de la respuesta a los riesgos informa periódicamente al director del proyecto sobre la eficacia del plan, cualquier efecto no anticipado o corrección necesaria para gestionar el riesgo adecuadamente. Este proceso incluye la actualización de los activos de los procesos de la organización y las bases de datos de lecciones aprendidas, para beneficio de futuros proyectos.

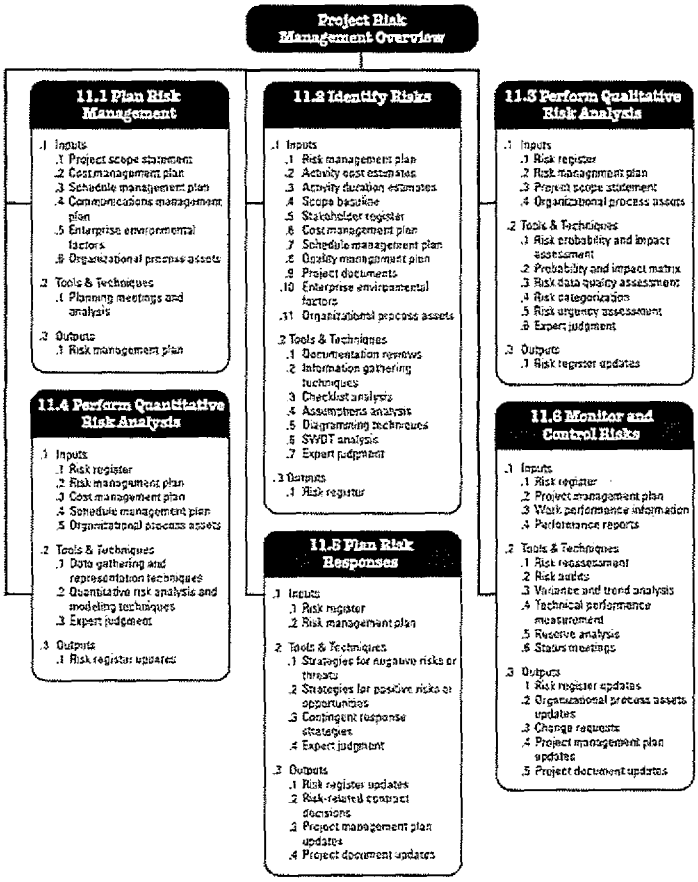

Figura 1: PMBOK (PMI)

Risk IT Framework de ISACA

Information Systems Audit & Control Association[5] (ISACA) es una organización sin fines de lucro, reconocida mundialmente como proveedor de conocimiento, emisor de certificaciones tales como Certified in Risk and Information Systems Control™ (CRISC™), apoyo y educación en seguridad y aseguramiento de sistemas de información, gobierno empresarial, administración de TI, al igual que riesgos y cumplimiento relacionados con TI. ISACA desarrolla y actualiza continuamente los marcos RISK IT®, COBIT y Val IT™, que ayudan a los profesionales de TI y los líderes de la empresa, a cumplir con sus responsabilidades de gobierno y entregar valor al negocio. COBIT, se encarga de gestionar todos los procesos relacionados con TI en la organización y tratar con eventos internos o externos a la misma. Los eventos internos pueden incluir los incidentes operacionales, los fracasos del proyecto, cambios de la estrategia de TI y las fusiones; y los eventos externos pueden incluir cambios en las condiciones del mercado, nuevos competidores, nuevas tecnologías disponibles y las nuevas regulaciones que la afectan. Estos eventos, promueven un riesgo y una oportunidad para evaluar el mismo y generar las soluciones oportunas. La dimensión del riesgo, y cómo gestionarlo, es el propósito principal de RISK IT [36].

RISK IT es un marco de referencia para la gestión de riesgos basado en el valor y los beneficios que la organización obtiene a través de las iniciativas de TI. A diferencia de otros modelos que se centran en eliminar los riesgos, RISK IT contempla la posibilidad de asumir riesgos que puedan traer beneficios a la organización, manteniendo un balance adecuado entre el Riesgo y el Valor para tomar ventaja de TI. *Los principios sobre los que se construye el marco RISK IT,* son los siguientes [37]:

- Estar siempre alineados con los objetivos de la organización.
- Permitir la alineación de riesgos relacionados con la gestión de TI con los riesgos de toda la organización.
- Balancear los costos y los beneficios en la gestión de riesgos.
- Promover una comunicación abierta y equitativa de los riesgos de TI.
- Establecer la definición y ejecución de las responsabilidades personales, para el funcionamiento dentro de los niveles de tolerancia aceptables y bien definidos.

[5] http://www.isaca.org

- Entender que se trata de un proceso continuo, que forma parte de las actividades diarias.

El marco de riesgos de TI se compone de nueve procesos, agrupados en tres dominios [36]:

I. *Gobierno del riesgo (RG)*

RG1 Establecer y mantener una visión común de riesgo: Garantizar que las tecnologías de la gestión de los riesgos de las prácticas están arraigadas en la empresa, lo que le permite garantizar una óptima rentabilidad ajustada al riesgo.

RG2 Integrar con la Gestión de riesgos de la empresa (ERM): Integrar la estrategia de TI y operaciones de riesgo con el riesgo de las decisiones estratégicas de negocios que se han realizado a nivel de empresa.

RG3 Tomar decisiones conscientes del riesgo de negocio: Asegurar que las decisiones de organización toman en cuenta toda la gama de oportunidades y las consecuencias de la dependencia de las TI para el éxito.

II. *Evaluación de riesgo (RE)*

RE1 Recoger datos: Identificar los datos pertinentes para hacer viable la identificación de riesgos de TI relacionados, el análisis y presentación de informes.

RE2 Analizar los riesgos: Desarrollar una información útil para apoyar las decisiones de riesgo que tenga en cuenta la importancia de factores de riesgo de negocios.

RE3 Mantener el Perfil de riesgo: Mantener actualizado el inventario completo de los riesgos conocidos y los atributos (por ejemplo, que se espera, la frecuencia de impacto potencial, disposición), los recursos, las capacidades y los controles como se entiende en el contexto de los productos empresariales, servicios y procesos.

III. *Respuesta de riesgo (RR)*

RR1 Articular el riesgo: Asegurar que la información sobre el verdadero estado de TI relacionados con la exposición y las oportunidades se disponga de manera oportuna y sea accesible a las personas adecuadas para dar respuestas apropiadas.

RR2 Gestionar los riesgos: Garantizar que las medidas para aprovechar las oportunidades estratégicas y la reducción del riesgo a un nivel aceptable se gestionan como una cartera.

RR3 Reaccionar a los eventos: Asegurar que las medidas para aprovechar las oportunidades inmediatas o limitar la magnitud de la pérdida de los acontecimientos relacionados con TI se activan en forma oportuna y eficaz.

Cada uno de los procesos anteriores, incluye una serie de actividades que deben ser realizadas por diversos roles de la empresa. Las organizaciones deben personalizar los componentes y las prácticas propuestas por el marco RISK IT, para adaptarlos a la organización y su contexto.

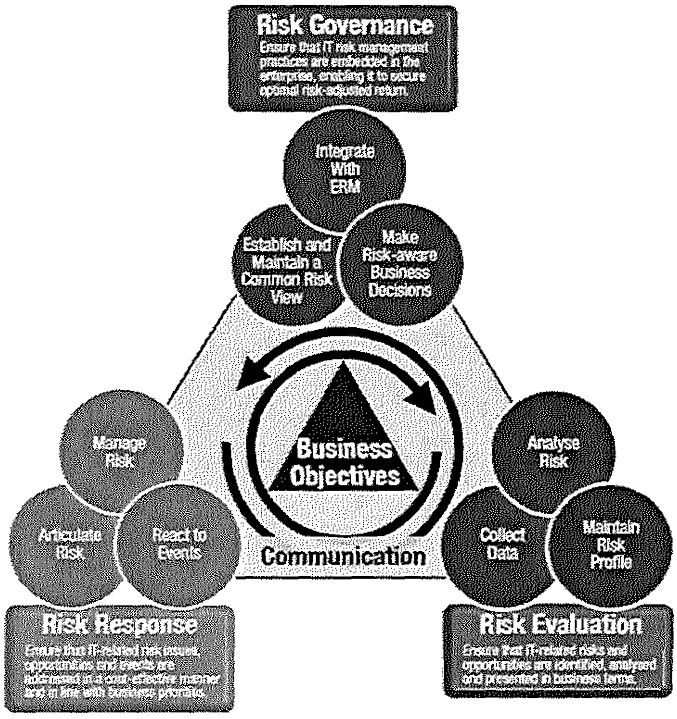

Figura 2: Risk IT Framework (ISACA)

ISO 31000:2009 Risk Management

La norma *ISO 31000:2009 denominada Risk Management – Principes and Guidelines*, por el *International Organization Standardization (ISO)*, tiene como objetivo ayudar a las organizaciones de todo tipo y tamaño a gestionar el riesgo con efectividad. Esta norma no es

específica a alguna industria o sector, puede ser utilizada por cualquier entidad pública, privada, organización sin fines de lucro, asociación, grupo o individuo.

La norma se estructura en tres elementos claves para una efectiva gestión de riesgos:

I. Principios:

La gestión de riesgos en una organización debe tener en cuenta los siguientes *principios* [56]:

- Crea y agrega valor.
- Está integrada en todos los procesos organizativos.
- Forma parte de la toma de decisiones.
- Aborda explícitamente la incertidumbre.
- Es sistemática, estructurada y oportuna.
- Está basada en la mejor información disponible.
- Es adaptable.
- Toma en cuenta los factores humanos y culturales.
- Es transparente e inclusiva.
- Es dinámica, iterativa y sensible al cambio.
- Facilita la mejora continua de la organización.

II. Framework:

El marco de trabajo sirve de guía en la gestión de riesgos efectiva a través de la aplicación del proceso de gestión de riesgos en diferentes niveles y dentro de contextos específicos de la organización [56]. El *framework para la gestión de riesgos* incluye varios pasos que se ilustran en la Figura 3.

- Mandato y compromiso.
- Diseño del marco de gestión de riesgos.
- La implementación de la gestión de riesgos.
- Monitoreo y revisión del marco de gestión de riesgos.
- Mejora continua del marco de gestión de riesgos.

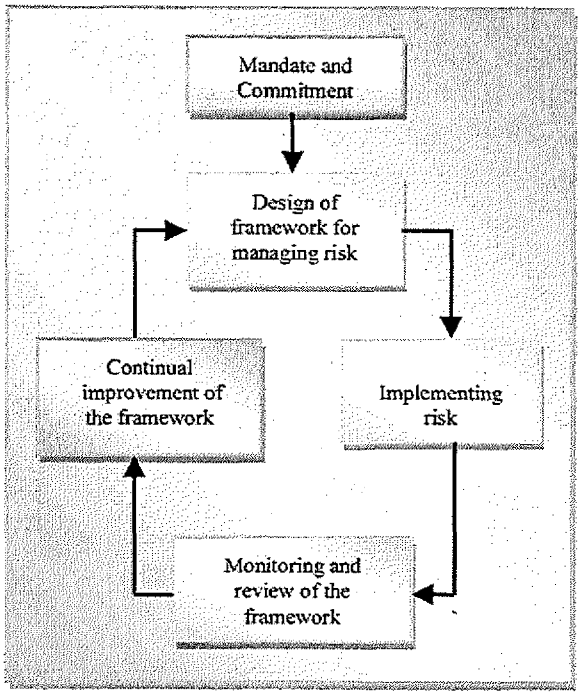

Figura 3: Risk management framework of ISO 31000 [57]

III. Proceso:

El *proceso de gestión de riesgos ISO 31000* adoptó en gran parte el proceso de la AS/NZS 4360:2004. El proceso de gestión de riesgos debe estar integrado a la gestión general, es decir debe ser parte de la cultura, de las mejores prácticas y del proceso de negocio de la organización [57].

En la figura 4 se representan *los componentes del proceso de gestión de riesgos propuestos en la norma ISO 31000,* que se mencionan a continuación [57]:

Comunicación y consultoría

- Análisis de los *stakeholders* y técnicas de comunicación.

Establecer el contexto

- Comprender el contexto organizacional.
- Taxonomía del riesgo tanto en ambientes internos como externos.

- Criterios de riesgo.

Valoración del Riesgo

- *Identificación de riesgos:* La profundización de las técnicas que han sido utilizadas: revisión de documentos, análisis de los *stakeholders*, estructura *breakdown* de riesgo, mapeo de los procesos de negocio.
- *Análisis y evaluación de riesgos:* Métodos cualitativos y métodos cuantitativos.

Tratamiento de Riesgos

- Estrategia de tratamiento de riesgos.
- Respuesta a la estrategia de emergencia y recuperación ante desastres.
- Construcción del plan de tratamiento de riesgos.
- Consideración de beneficios y costos.

Monitoreo y revisión

- Determinación de quién está haciendo el seguimiento y revisión.
- Qué necesita ser monitoreado y revisado.
- Qué información necesita ser evaluada.
- Procedimientos a utilizar.
- Proceso de información.

Documentación del proceso de gestión de riesgos

- Registro de cada estado del proceso.
- Almacenamiento de documentos.
- Utilizar técnicas de gestión del conocimiento.

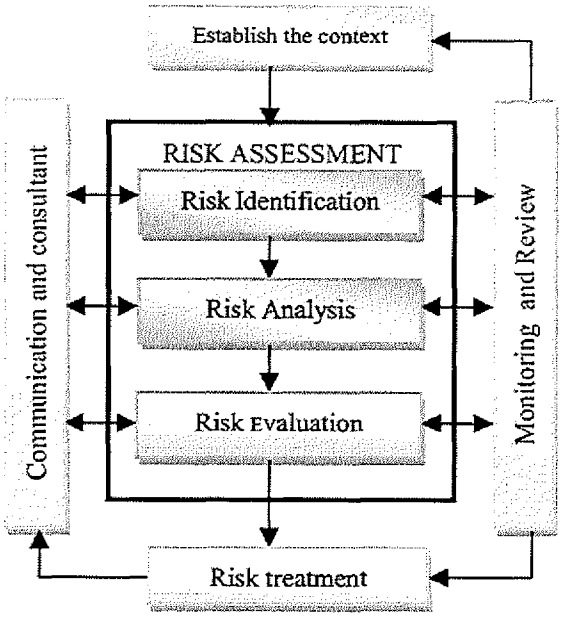

Figura 4: Risk management process of ISO 31 000 [57]

Modelos específicos de la Ingeniería de Software

Entre los modelos de gestión de riesgos aplicados específicamente en la Ingeniería de software, se pueden mencionar: *Software Risk Management (SRM), Team Risk Management (TRM) y CMMI (Capability Maturity Model Integration*) propuestos por el Software Engineering Institute (SEI), y las normas *IEEE 1540:2001* e *ISO/IEC 16085:2006* desarrolladas por el Institute of Electrical and Electronics Engineers, e International Organization Standardization (ISO), International Electrotechnical Commission (IEC), respectivamente. Dichos modelos y normas, se presentan brevemente a continuación.

Software Risk Management (SRM)

Es un marco para la gestión de riesgos desarrollado en el Instituto de Ingeniería de Software (SEI) [33]:

"El objetivo de la Gestión de Riesgos es permitir a los ingenieros, gerentes y otros tomadores de decisiones identificar con la suficiente antelación, los riesgos asociados con la adquisición, desarrollo, integración y despliegue de software, de modo que las estrategias de gestión y mitigación apropiadas se puedan desarrollar en forma oportuna".

Los *principios en los que se basa la gestión de riesgos* de este modelo son los siguientes [33]:

Comunicación abierta:

La gestión eficaz del riesgo requiere una atención constante a la promoción del principio básico de la libre la comunicación:

- fomentar que la información fluya libremente en y entre todos los niveles del proyecto,
- permitir la comunicación formal, informal y espontánea,
- procesos consensuados que valoran la voz individual, pueden aportar conocimiento para la identificación y gestión de riesgos.

Gestión integrada:

Este principio ayuda a asegurar que los procesos de gestión de riesgos, la documentación y disciplina están en consonancia con la cultura del proyecto, e integrados a la rutina del proyecto. La gestión integrada exige:

- gestión de riesgo como una parte integral y vital de la gestión de proyectos,
- adaptación de los métodos y herramientas de gestión de riesgos a la infraestructura de un proyecto y la cultura.

Trabajo en equipo:

La gestión del riesgo requiere que los miembros del proyecto trabajen juntos en encontrar y analizar los potenciales riesgos a los que se enfrentan. El trabajo en equipo requiere:

- trabajar en conjunto para lograr un objetivo común,
- habilidades, conocimiento y talento del grupo.

Proceso continuo:

El Proceso continuo de gestión de riesgos requiere:

- mantener una vigilancia constante,
- identificar y gestionar los riesgos de forma rutinaria en todas las fases del ciclo de vida del proyecto.

Visión a futuro:

Este principio desarrolla la capacidad de mirar hacia adelante, el impacto de las decisiones actuales sobre las opciones futuras. Visión prospectiva requiere:

- pensar hacia el mañana, identificar las incertidumbres, previendo los posibles resultados,
- gestionar los recursos y actividades del proyecto, anticipándose a las incertidumbres.

Perspectiva global:

Este principio requiere que el personal del proyecto reemplace sus intereses y puntos de vista con los que se benefician del bien común de todo el proyecto, y el cliente armonice sus perspectivas con las de la empresa. El personal del proyecto debe desarrollar y compartir una visión común, y ser capaz de hacer frente y mitigar los riesgos específicos de forma conjunta. El principio de perspectiva global requiere:

- visualizar el contexto de desarrollo de software a un nivel más grande de definición, diseño y desarrollo de sistemas,
- reconocer tanto el valor potencial de las oportunidades y el impacto potencial de efectos adversos.

Visión compartida del producto:

Una vez definida la visión compartida del producto, se hace mucho más fácil llegar a un entendimiento común de lo que pueda afectar adversamente el costo o características del resultado final. La visión compartida del producto requiere:

- compartir una visión del producto en base a un propósito común, la propiedad compartida y el compromiso colectivo,
- enfocar la atención en los resultados.

El paradigma de gestión de riesgos muestra las diferentes actividades relacionadas con la gestión de riesgos asociados con el desarrollo de software, es representada por un círculo para hacer hincapié en que la gestión de riesgos es un proceso continuo, mientras que las flechas muestran el flujo lógico de información entre las actividades. Comunicación se coloca en el centro del paradigma, siendo el conducto a través del cual fluye toda la información y, a menudo, el obstáculo más grande en la gestión del riesgo. *Las actividades del paradigma de gestión de riesgos* se describen a continuación [33]:

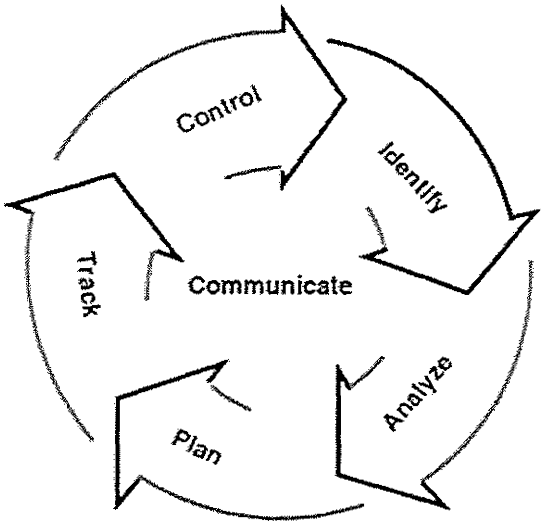

Figura 5: Software Risk Management (SEI)

Identificación:

Antes de que los riesgos se conviertan en problemas, deben ser identificados. El SEI ha desarrollado técnicas que alientan al personal del proyecto a plantear sus inquietudes y problemas.

Análisis:

Es la conversión de datos en información para la toma de decisiones del riesgo. Esta actividad suministra las bases al director de proyecto para trabajar en los riesgos correctos y más críticos.

Planificación:

Consiste en transformar la información del riesgo en decisiones y acciones para hacer frente a los riesgos individuales, priorizando las acciones de riesgos y creando un plan integrado de gestión de riesgos que puede adoptar muchas formas:

- Mitigar el impacto del riesgo mediante un plan de contingencia para cuando ocurra el riesgo.
- Evitar el riesgo cambiando el diseño del producto o el proceso de desarrollo.
- Aceptar el riesgo y sus consecuencias si se produce.

- Estudiar el riesgo adquiriendo mayor información y determinar sus características para tomar mejores decisiones.

La clave para planificar las acciones de riesgos es tener en cuenta las consecuencias futuras de una decisión tomada hoy.

Seguimiento:

Se basa en supervisar el estado de los riesgos y de las medidas adoptadas para mejorarlos. Métricas son monitoreadas para permitir la evaluación de la situación de los riesgos y de los planes de mitigación.

Control:

El control de riesgos corrige las desviaciones de las acciones de riesgo planeadas. El control de riesgos se funde en la gestión de proyectos, y se basa en los procesos de gestión de proyectos para el control de los planes de acción de riesgo, corrige las variaciones de planes, responde a la activación de eventos y mejora el ambiente de los procesos de gestión de riesgos.

Comunicación:

La comunicación de riesgos se encuentra en el centro del modelo para enfatizar tanto en su extensión y en su criticidad, como parte integral de todas las actividades de gestión de riesgos y entre los niveles dentro del proyecto de desarrollo y organización. Si no existe una comunicación efectiva ningún enfoque de gestión de riesgos puede ser viable.

Team Risk Management (TRM)

Los riesgos de software se encuentran entre los riesgos menos medidos o gestionados en un proyecto. Hay muchos desafíos en la aplicación del proceso de gestión de riesgos de equipo dentro de una organización, el Programa de Riesgo en el SEI ha ido madurando sus métodos de gestión de riesgos mediante la aplicación de los mismos a los proyectos de desarrollo de software en el ámbito del gobierno y de la industria. La gestión de riesgos de equipo ofrece al cliente y al proveedor, los procesos, métodos y herramientas que permitan a ambas organizaciones, individual y conjuntamente, ser cada vez más anticipatorias en los procesos de toma de decisiones de la gestión de riesgos en los proyectos de desarrollo de software. Este enfoque se basa en una visión compartida del proyecto que se construye de las opiniones individuales del cliente y del proveedor, en una comunicación abierta y en un trabajo cooperativo para manejar los riesgos antes de que se conviertan en problemas [34].

La gestión de riesgos de equipo se basa en los siguientes principios [34]:

Visión compartida del producto: fundado en los propósitos de la comunidad, la propiedad compartida y el compromiso colectivo.

Búsqueda de incertidumbres: pensar en el mañana anticipándose a los posibles resultados, mediante la identificación y reconocimiento de la incertidumbre.

Comunicación abierta: el libre flujo de información en y entre todos los niveles del proyecto a través de la comunicación formal, informal y espontánea, y procesos basados en el consenso.

Valor de la percepción individual: la voz individual que puede aportar un conocimiento único y los conocimientos para la identificación y gestión de riesgos.

Perspectiva de Sistema: desarrollo de software visible dentro de la mayor definición, diseño y desarrollo a nivel de sistemas.

Gestión integral del proyecto: la gestión del riesgo como una parte integral y vital de la gestión del proyecto.

Estrategias proactivas: que involucran las actividades de planificación y ejecución en base a anticipar eventos futuros.

Metodología sistemática y adaptable: un enfoque sistemático que se adapta a la infraestructura y cultura del proyecto.

Procesos continuos y rutinarios: una vigilancia continua que se caracteriza por las actividades de identificación y gestión de riesgos de rutina en todas las fases del ciclo de vida del proyecto.

La gestión de riesgos de equipo extiende el paradigma de gestión de riesgos SEI y se implementa como parte de la rutina, continua e integral de las actividades de gestión de proyectos. La gestión de riesgos de equipo incluye procesos continuos y una actividad inicial de una sola vez, la evaluación de riesgos de la línea de base. Los riesgos identificados en la línea de base representan los más importantes a los que se enfrenta el proyecto y son el resultado del esfuerzo de articular la gestión de "equipo" en las organizaciones asociadas cliente y proveedor. Cada organización gestiona sus riesgos a través de las actividades de procesos continuos [34].

Los procesos continuos incorporan los cinco pasos del paradigma SEI en cuatro procesos, mediante la combinación de los pasos de identificación y análisis de riesgos en el proceso de identificación y análisis de riesgos de rutina. A excepción de la revisión de equipo, todas las actividades continuas de gestión de riesgos de equipo están incluidas en un ciclo básico de procesos, que comienza con la identificación de riesgos de rutina y análisis para identificar nuevos riesgos, le procede la planificación de estrategias y acciones específicas para hacer frente a los riesgos. Los pasos siguientes del proceso de seguimiento y control monitorean los resultados de los planes de acción y controlan el progreso de las actividades de gestión de riesgos del proyecto. Todas las actividades están vinculadas entre sí a través de los procesos de comunicación formal e informal. Estas actividades mejoran las interacciones de cooperación, la confianza entre los socios y miembros del equipo, construyen y refuerzan la visión compartida del proyecto requerido para la gestión eficaz de riesgos equipo. El proceso de revisión de equipo es un proceso inter-organizacional realizado conjuntamente entre el gobierno y el contratista, los otros procesos continuos de gestión de riesgos implementan procesos intra-organizacionales e inter-organizacionales [34].

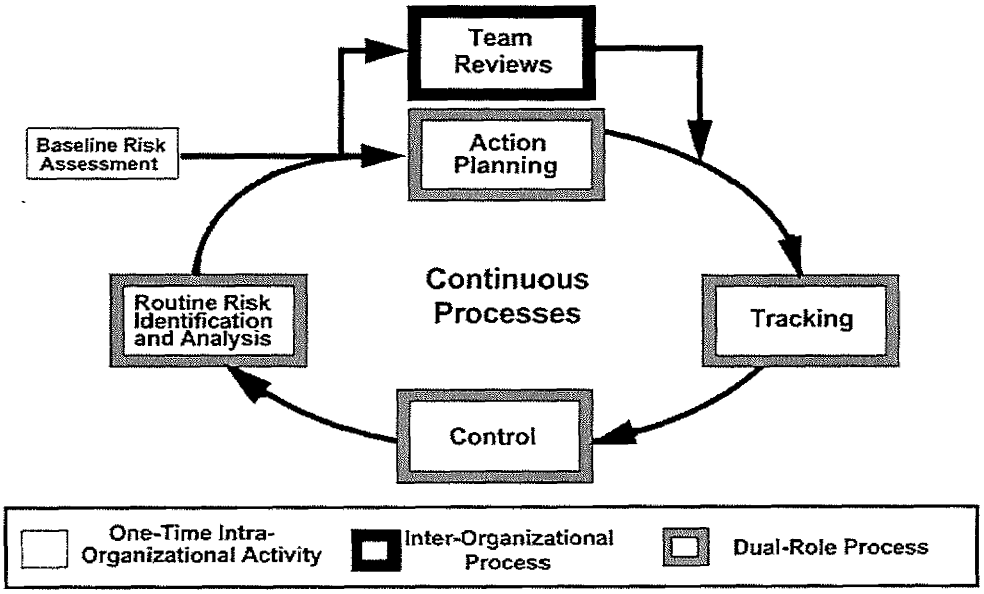

Figura 6: Team Risk Management (SEI)

CMMI (Capability Maturity Model Integration)

Los modelos CMMI ayudan a las organizaciones a mejorar sus procesos. Son desarrollados por equipos expertos internacionales, con miembros procedentes de la industria, del gobierno y del Software Engineering Institute (SEI). Una de las áreas de interés que cubre el CMMI es la de desarrollo de productos y servicios CMMI-DEV. Este modelo suministra una orientación para aplicar las buenas prácticas CMMI en las actividades para desarrollar productos y servicios de calidad en una organización de desarrollo, con el fin de cumplir las necesidades de clientes y usuarios finales. Según el informe del SEI publicado en septiembre de 2011, "CMMI® for Development SCAMPISM Class A Appraisal Results 2011 Mid-Year Update" son casi 200 las organizaciones españolas actualmente en posesión del certificado CMMI, mientras que esta cifra era de menos de 10 en 2005, y ocupan unas posiciones muy destacadas países como Brasil, con 125 organizaciones certificadas, México con 95 y Argentina con 66 entidades [35].

Entre alguna de las mejoras significativas que se pueden encontrar en CMMI-DEV en su versión 1.3, es la corrección de las prácticas de ingeniería para reflejar las buenas prácticas del sector y añadir orientaciones para las organizaciones que utilicen métodos ágiles [35].

CMMI da soporte a dos caminos de mejora usando niveles, que corresponden a las dos aproximaciones de mejora de procesos denominadas "representación continua y representación por etapas". La representación continua utiliza los "niveles de capacidad" para caracterizar el estado de los procesos de la organización con respecto a un área de proceso individual. La representación por etapas utiliza los "niveles de madurez" para caracterizar el estado global de los procesos de la organización con respecto al modelo como un todo [35].

Los niveles de capacidad se refieren a la consecución de la mejora de procesos de una organización en áreas de proceso individuales, y constituyen un medio para mejorar de forma incremental los procesos que corresponden a un área de proceso determinada. Los cuatro niveles de capacidad se numeran del 0 al 3 y corresponden a: incompleto, realizado, gestionado y definido, respectivamente. Los niveles de madurez se refieren a la consecución de la mejora de procesos de una organización en múltiples áreas de proceso, y constituyen un medio para mejorar los procesos correspondientes a un conjunto determinado de áreas de proceso (es decir, nivel de madurez). Los cinco niveles de madurez se numeran del 1 al 5 y corresponden a: inicial, gestionado, definido, gestionado cuantitativamente y en optimización, respectivamente [35].

CMMI define *"un área de proceso como un grupo de prácticas relacionadas dentro de un área que, cuando se implementan conjuntamente satisface un conjunto de metas consideradas*

importantes para mejorar esa área" [35]. La Gestión de Riesgos (RSKM) es un área de proceso de gestión de proyectos en el nivel de madurez 3.

La *Gestión de Riesgos (RSKM) es un proceso continuo,* que describe una evolución de las prácticas específicas para planificar, prevenir y mitigar los riesgos sistemáticamente, con el propósito de identificar proactivamente los potenciales problemas antes de que ocurran, planificar e invocar las actividades de tratamiento de riesgos cuando sean necesarias a lo largo del ciclo de vida del producto o proyecto, para mitigar los impactos adversos sobre la consecución de los objetivos de proyecto [35].

El área de proceso Gestión de Riesgos (RSKM) propone alcanzar tres metas, cada una de las cuales incluye prácticas específicas [35]:

Preparar la gestión de riesgos: establecer y mantener una estrategia para identificar, analizar y mitigar los riesgos, documentado en un plan de gestión de riesgos. La estrategia por lo general incluye la identificación de las fuentes de riesgos, el esquema utilizado para clasificar los riesgos y los parámetros usados para evaluar, limitar y controlar los riesgos para su tratamiento eficaz.

Incluye las siguientes prácticas específicas:

- Determinar las fuentes y las categorías de los riesgos.
- Definir los parámetros usados para analizar y clasificar los riesgos, y para controlar el esfuerzo de la gestión de riesgos.
- Establecer y mantener la estrategia que se utilizará para la gestión de riesgos.

Identificar y analizar los riesgos: para determinar su importancia relativa. El análisis de riesgos consiste en la identificación de los riesgos de las fuentes internas y externas, y la evaluación de cada uno de los riesgos anteriores para determinar su probabilidad y consecuencias. La clasificación de los riesgos, basada en una evaluación frente a las categorías de riesgos establecidas y los criterios desarrollados para la estrategia de gestión de riesgos, suministra información necesaria para su tratamiento.

Incluye las siguientes prácticas específicas:

- Identificar y documentar los riesgos.
- Evaluar y clasificar cada riesgo identificado usando las categorías y los parámetros definidos para el riesgo, y determinar su prioridad relativa.

Mitigar los riesgos: para reducir los impactos adversos sobre la obtención de los objetivos. Incluye desarrollar opciones de tratamiento de riesgos, monitorizar los riesgos y realizar las actividades de tratamiento de riesgos establecidos en la estrategia de gestión de riesgos cuando se sobrepasan los umbrales definidos. Los planes de mitigación de riesgos se desarrollan e implementan para los riesgos seleccionados a fin de reducir de forma proactiva el impacto potencial de la ocurrencia del riesgo, también puede incluir planes de contingencia para tratar con el impacto de los riesgos que puedan ocurrir a pesar de los intentos por mitigarlos.

Incluye las siguientes prácticas específicas:

- Desarrollar un plan de mitigación de riesgos en concordancia con la estrategia de gestión de riesgos.
- Monitorizar el estado de cada riesgo periódicamente e implementar el plan de mitigación de riesgos según sea apropiado.

IEEE 1540:2001 Ciclo de Vida del Software – Gestión de riesgos

La gestión del riesgo permite determinar continuamente la viabilidad de los planes del proyecto de software, mejorar la búsqueda e identificación de posibles inconvenientes que puedan afectar las actividades del ciclo de vida del software y la calidad del producto software.

El estándar IEEE 1540 de gestión de riesgos de software es compatible con la adquisición, suministro, desarrollo, operación y mantenimiento de productos y servicios de software [55]. Dicho estándar está diseñado de manera que se pueda aplicar de forma independiente o conjuntamente con la estándar IEEE 12207, que es una serie de estándares de procesos del ciclo de vida de software. *IEEE 12207 reconoce a la gestión activa del riesgo como un factor clave de éxito en la gestión de un proyecto de software* [55].

El estándar IEEE 1540 se puede adaptar para su aplicación a todos los proyectos apropiados de una organización o para su uso en un proyecto individual. Si bien esta norma está orientada a la gestión del riesgo en proyectos de software, también puede ser útil para la gestión de riesgos tanto a nivel de sistema y a nivel de organización. *La norma define un proceso de gestión de riesgos de software que se compone de varias actividades y tareas que funcionan de manera iterativa y que deben integrarse con otras prácticas y sistemas de gestión de riesgos de la organización* [55]:

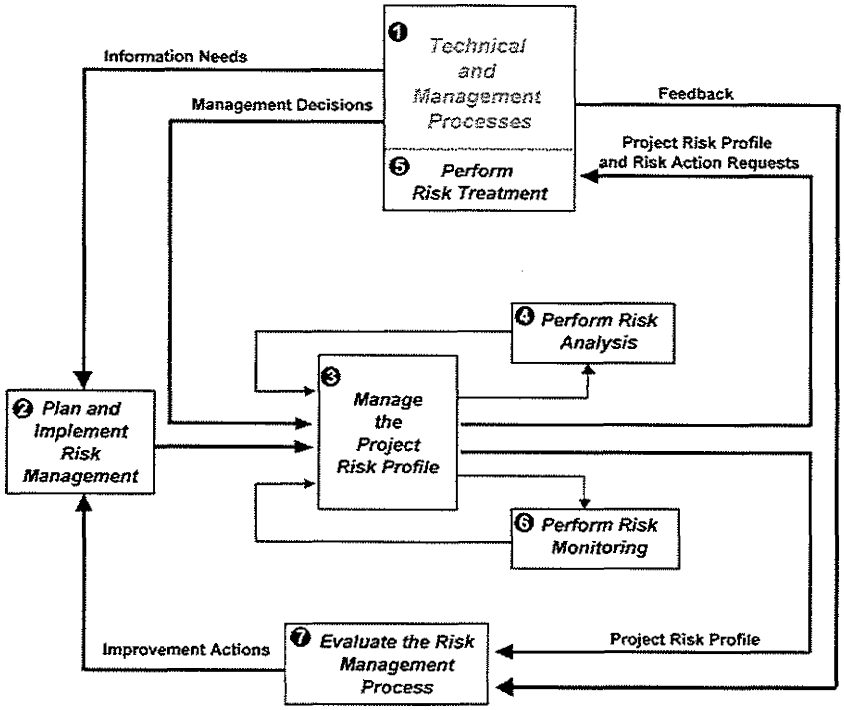

Figura 7: Risk management process model [55]

1. *Procesos técnicos y de gestión:* relacionados con los grupos de interés o stakeholders que definen los requisitos de información que el proceso de gestión de riesgos debe soportar. Estos requisitos se transmiten tanto a "*Planificar e Implementar la gestión de riesgos*" y "*Gestionar el perfil de riesgo del proyecto*".

2. *Planificar e Implementar la gestión de riesgos:* establece un proceso de gestión de riesgos de software que debe estar alineado al proceso de gestión de riesgos de la organización, para ello se deberá definir quién lo llevará a cabo, el proceso específico a utilizar, asignar los recursos necesarios, y definir como los riesgos y su tratamiento deben ser comunicados y coordinados entre los stakeholders.

3. *Gestionar el perfil de riesgo del proyecto:* consiste en reunir información sobre el contexto actual e histórico de la gestión de riesgos y el estado de los riesgos. Dicha información se actualiza a través del análisis de riesgos. El perfil de riesgo del proyecto se mantiene a lo largo del ciclo de vida del software.

4. *Realizar análisis de riesgos:* consiste en identificar los sucesos iniciadores, amenazas y situaciones que crean riesgos, estimar su probabilidad de ocurrencia y consecuencias, y preparar recomendaciones de tratamiento de los riesgos en base a un análisis de orden de prioridad de riesgo que se lleva a cabo en forma continua a lo largo del ciclo de vida del software. Las recomendaciones de tratamiento, junto con el estado de otros riesgos y su estado de tratamiento se envían a la dirección para su revisión.

5. *Realizar el tratamiento de riesgos:* implica la selección, la planificación, el seguimiento y el control de las acciones para reducir el nivel de exposición de los riesgos considerados no aceptables por los stakeholders. Se crean los planes de tratamiento para los riesgos que están por encima del umbral de riesgos, dichos planes son coordinados con otros planes de gestión y otras actividades en curso.

6. *Realizar el monitoreo de riesgos:* incluye evaluar la efectividad del tratamiento de riesgos, revisar y actualizar los estados de riesgos individuales y el contexto de gestión de riesgos, y buscar continuamente nuevos riesgos.

7. *Evaluar el proceso de gestión de riesgos:* consiste en suministrar información a las partes interesadas sobre la calidad del proceso de gestión de riesgos e identificar las oportunidades para mejorar los procedimientos, procesos o políticas de gestión de riesgos de la organización para reducir o eliminar los riesgos sistémicos. Se requiere una evaluación periódica del proceso de gestión de riesgos para garantizar su eficiencia, las mejoras definidas como resultados de dicha evaluación se implementan en la actividad de planificación e implementación de la gestión de riesgos.

Estándar Internacional ISO/IEC 16085:2006

El estándar ISO/IEC/IEEE 16085:2004, sustituyó al estándar IEEE 1540. Como resultado de la revisión de los estándares anteriores, se crea el estándar ISO/IEC 16085:2006 que define el mismo proceso continuo de gestión de riesgos que su primera versión y aborda sistemáticamente el riesgo en todo el ciclo de vida, incluyendo el suministro, desarrollo, operaciones, mantenimiento y adquisición de software.

Puede aplicarse conjuntamente con las norma: ISO/IEC 12207, ISO/ IEC 15288, o de forma independiente. El estándar ISO/IEC 16085 es ampliamente compatible con el proceso de gestión de riesgos documentado en la norma ISO/IEC 15288 y suministra información adicional del procedimiento para ayudar a la planificación y ejecución de la gestión de riesgos. Cuando el

estándar ISO/IEC 16085 se utiliza con el estándar ISO/IEC 12207, supone que los otros procesos y técnicas de gestión de la ISO / IEC 12207 realizan el tratamiento del riesgo.

La gestión de riesgos es más eficaz cuando se integra a un proceso de medición. ISO/IEC 15939, define un proceso de medición aplicable a todas las disciplinas de ingeniería y gestión. El proceso de medición definido en la ISO / IEC 15939 trabaja en conjunto con las actividades y tareas de gestión de riesgos establecidas en el estándar ISO/IEC 16.085 para ayudar a caracterizar y cuantificar los riesgos.

CAPÍTULO 3: AGILIDAD EN EL DESARROLLO DE SOFTWARE

Ingeniería de Software y Agilidad

Desde la denominada *"crisis del software"*, el desarrollo de software ha evolucionado notablemente con la importación de prácticas y metodologías existentes en otras disciplinas, mejorando sus procesos y dando un giro importante a la industria del software.

"La agilidad es la capacidad de responder a las necesidades del negocio para mantenerse en un entorno empresarial inestable y cambiante" [38]. La agilidad está presente en diferentes disciplinas. En muchas industrias frente a cambios impredecibles en la demanda de productos y necesidades del cliente se requieren capacidades competentes en el cambio. A principios de 1990, *el concepto de fabricación ágil (AM) fue utilizado para describir una capacidad corporativa para una rápida adaptación a las necesidades cambiantes, tales como la intensificación de la competencia mundial, la reducción en el tiempo de espera y la esperanza de vida de los productos, la diversificación de la demanda y las nuevas tecnologías* [45]. La producción de software opera en entornos inciertos y dinámicos, enfrentándose actualmente a muchos problemas y desafíos similares.

Es necesario extender la visión de la ingeniería de software hacia la adopción de las principales enseñanzas aplicables en otras áreas, incorporar nuevas mejoras en la industria de desarrollo de software considerando la agilidad como un concepto de negocio más amplio orientado a la organización, tras el aprendizaje de la fabricación ágil (AM) y el desarrollo de nuevos productos (NPD) [45]. Si los comparamos con la producción de software, éste tiene ciertas características intrínsecas: ciclos de desarrollo (tiempo de cambio) más rápidos, por lo general no hay limitaciones físicas de re-trabajo (costo del cambio) y permite rediseños radicales incluso en una etapa tardía (magnitud del cambio). Los modelos actuales de desarrollo ágil de software abordan sólo algunos de los principios de fabricación ágil (AM), siendo uno de sus principales retos ampliar el ámbito de aplicación de ingeniería hacia la dimensión organizativa. Un enfoque notable en esta dirección es Dynamic Systems Development Methodology (DSDM) que si bien se origina casi al mismo tiempo que fabricación ágil (AM), no se considera como corriente principal del desarrollo ágil de software [45].

Se han intentado explicar las ideas centrales de *agilidad* en el desarrollo de software, conocidas en otros campos como *flexibilidad* y *delgadez,* examinando las tendencias similares en otras disciplinas [41]: *la fabricación ágil* mediante la integración de las relaciones cliente-proveedor, la gestión del cambio, la incertidumbre, la complejidad, la utilización de los recursos humanos, y la información; y *el desarrollo Lean* que persigue alcanzar la calidad

mediante la eliminación de residuos y tiene sus orígenes en el sistema de producción de Toyota a partir de la década de 1950.

Ser ágil significa "entregar rápidamente, cambiar frecuentemente" [43]. "La novedad de los métodos ágiles no son las prácticas que se utilizan, sino el reconocimiento de las personas como los principales motores de éxito del proyecto, junto con un intenso enfoque en la eficacia y manejo del proceso" [43].

La agilidad parece tener sus orígenes en el dominio de la fabricación y la gestión. Los principios y prácticas ágiles no son nuevos en la comunidad de software, la novedad es la forma en que fueron expuestos juntos en un marco teórico y práctico [40]. *"Los principios no son una definición oficial de la agilidad, sino que son las directrices para la entrega de software de alta calidad de una manera ágil"* [40].

"Ser ágil es más que simplemente seguir las directrices que se supone debe hacer un proyecto ágil, la verdadera agilidad es más que una colección de prácticas, es un estado de ánimo. Los procesos pueden tener un aspecto ágil pero no sentirse ágiles" [43]. La adopción ágil para ser exitosa, requiere la integración conceptual entre organización y visión ágil. Ser ágil es una cuestión cultural, si ésta no existe entonces la organización no puede ser ágil. Además, la cultura debe ser de apoyo de la negociación, así como la negociación forma parte de la cultura ágil.

Manifiesto Ágil

Por mucho tiempo se han discutido soluciones en el ámbito de la ingeniería de software que abarcan desde la estandarización y medición del proceso de software hasta una cantidad de herramientas y prácticas específicas. Muchos profesionales experimentados han sugerido mejoras en las prácticas y métodos de desarrollo de software, que representan una desviación importante de los enfoques tradicionales. En esta línea, el movimiento ágil está cambiando las formas en que se desarrolla software en todo el mundo.

Después de casi una década de esfuerzos aislados, en febrero de 2001 en Utah-EEUU, se reunieron 17 empresarios de la industria del software[6] y como resultado del debate respecto a las metodologías, principios y valores que deben regir el desarrollo de software de buena calidad, en tiempos cortos y flexible a los cambios, se aceptó el término *ágil* para hacer

[6] Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland y Dave Thomas.

referencia a nuevos enfoques metodológicos en el desarrollo de software. En esta reunión se creó *"The Agile Alliance"*, organización sin ánimo de lucro dedicada a promover los conceptos relacionados con el desarrollo ágil de software y acompañar a las organizaciones para que adopten dichos conceptos. Se redacta un documento denominado *Manifiesto Ágil* que resume en cuatro valores y doce principios las mejores prácticas de desarrollo de software [48].

El manifiesto ágil introdujo notables transformaciones, los diferentes métodos se esforzaron por adherirse a los principios básicos del mismo. En un primer lugar, emerge un movimiento distinto hacia el trabajo colaborativo otorgando privilegios a los procesos que anteriormente eran restringidos. En segundo lugar, la mentalidad dominante Lean reduce al mínimo el trabajo innecesario y mantiene sólo la documentación esencial. En tercer lugar, los clientes participan activamente en el desarrollo, guiando la evolución del producto software o servicio final. En cuarto lugar, el reconocimiento de que la incertidumbre es una parte integral del desarrollo de software y la tendencia a controlar las desviaciones por medios estadísticos u otros resultaba inútil [40].

Los primeros años del Manifiesto Ágil se vieron marcados por la adhesión de unos pocos a los principios y el escepticismo de muchos [40]. Los diez años siguientes a la emisión de dicho manifiesto, las investigaciones revelan que la comunidad de software ha dedicado una gran atención al desarrollo ágil evidenciada en el número de publicaciones en los diversos foros científicos. El desarrollo ágil de software ha sido un tema de investigación en todos los continentes en un total de sesenta y tres (63) de países, ocupando Argentina la posición veintiséis (26). *Tras el impulso inicial de los estudios sobre la programación extrema, la comunidad académica parece haber centrado su atención en Scrum* [40].

Valores para el Desarrollo Ágil de Software

Los participantes de Agile Alliance, encontraron consenso en torno a cuatro valores principales que exponen en el Manifiesto Ágil[7]:

"Estamos descubriendo mejores formas de desarrollar software haciéndolo y ayudando a que otros lo hagan. A través de este trabajo hemos llegado a valorar:

Individuos e interacciones del equipo de desarrollo sobre *procesos y herramientas.*

Desarrollar Software por encima de *una completa documentación.*

[7] http://www.agilemanifesto.org

La colaboración con el cliente por encima de *la negociación del contrato.*

Responder a los cambios más que *seguir estrictamente un plan".*

Principios Inspirados en los Valores Ágiles

Doce principios[8] complementan el Manifiesto Ágil, y explican aún más lo que es *ser ágil*:

1. *La máxima prioridad es satisfacer al cliente a través de la entrega temprana y continua de software valioso.*
2. *Bienvenidos requisitos cambiantes, incluso tarde en el desarrollo. Procesos ágiles aprovechan el cambio para obtener ventajas competitivas del cliente.*
3. *Entregar software en un trabajo con frecuencia de un par de semanas a un par de meses, con una preferencia a la escala de tiempo más corta.*
4. *La gente de negocios y desarrolladores deben trabajar juntos todos los días durante todo el proyecto.*
5. *Construir proyectos alrededor de individuos motivados. Darles el ambiente y el apoyo que necesitan, y confiar en ellos para hacer el trabajo.*
6. *El método más eficiente y eficaz de transmisión de información hacia y dentro de un equipo de desarrollo es la comunicación cara a cara.*
7. *El desarrollo de software es la principal medida de progreso.*
8. *Procesos ágiles promueven el desarrollo sostenible. Los patrocinadores, los desarrolladores y los usuarios deben ser capaces de mantener un ritmo constante de forma indefinida.*
9. *La atención continua a la excelencia técnica y el buen diseño mejora la agilidad.*
10. *La Simplicidad es esencial.*
11. *Las mejores arquitecturas, requerimientos y diseños emergen de equipos auto organizados.*

[8] http://www.agilealliance.org

12. *A intervalos regulares, el equipo reflexiona sobre cómo ser más eficaz y ajusta su comportamiento en consecuencia.*

El desafío de la industria de software: Ser o no ser ágil

La historia detrás de los métodos ágiles, deja de manifiesto que los modelos tradicionales de desarrollo de software comenzaron a plantearse dificultades para dar cabida al cambio, los usuarios cambiaban de opinión sobre la marcha del proyecto sin estar seguros de lo que querían, y al presentarse el cambio se hacía difícil detener el impulso del proyecto para dar lugar al mismo.

El desarrollo ágil evolucionó de las experiencias personales y la sabiduría colectiva de los consultores y líderes de la comunidad de software. La mayoría de las prácticas ágiles se concibieron inicialmente como prácticas individuales e intuitivas, que se fueron naturalizando en hábitos. Por lo tanto, había una necesidad de fundamentos teóricos o apoyo empírico que arrojaran luz sobre los beneficios de estas prácticas y como sus interacciones podrían producir resultados valiosos [40].

Una revisión sistemática publicada hasta el año 2005, demostró que la fuerza de la evidencia de los estudios empíricos era muy baja con respecto a los beneficios y limitaciones de los métodos ágiles para tomar una decisión con respecto a su adopción. Además, la atención de la investigación se centraba casi exclusivamente en la Programación Extrema, y muy poco en Scrum en comparación con su popularidad en la industria de software. El hallazgo de la revisión sobre la necesidad de aumentar el número y la calidad de los estudios sobre el desarrollo ágil de software, y destinar más recursos a la investigación de las prácticas en los equipos maduros, fue consistente con las críticas en relación con el escaso apoyo científico para muchas de las afirmaciones hechas por la comunidad ágil [41].

De los resultados de una encuesta realizada en la industria de software en Irlanda del Norte en 2010 y 2012, dirigida a comprender el progreso del enfoque ágil de desarrollo de software, surgen dos observaciones principales [39]: *el nivel de adopción del paradigma ágil es alto y creciente, sustituirá progresivamente al modelo de cascada tradicional como el enfoque estándar de desarrollo de software;* y *es necesario identificar los beneficios y las limitaciones del desarrollo ágil para ayudar a las empresas a optimizar su uso.*

Varias lecciones se pueden aprender de la discusión de expertos de todo el mundo, cuando se evalúa la posibilidad de aplicar métodos ágiles a un proyecto. El factor más importante que determina cuando ágil es aplicable, es probablemente el tamaño del equipo. Cuando éste

crece la coordinación de las interfaces se convierte en un tema dominante y la comunicación cara a cara se vuelve compleja y difícil [43].

Otra decisión relevante es consumir grandes esfuerzos en la planificación por adelantado y evitar el cambio, o planificar con menos rigor y abrazar el cambio. El enfoque ágil es más apropiado cuando los requerimientos son emergentes y en rápida evolución, y existe algo de incertidumbre técnica. Se sugiere el uso de métodos tradicionales para proyectos donde los requerimientos cambian menos de 1% por mes. Los procesos impulsados por el plan son en su mayoría necesarios en software de alta seguridad, donde los objetivos tradicionales como: predictibilidad, repetitividad y optimización, son características de seguridad fiable en el desarrollo de software crítico [43].

Los métodos ágiles requieren menos entrenamiento formal que los métodos tradicionales, el énfasis está en el desarrollo de habilidades más que en el aprendizaje del método. La pericia en la construcción de sistemas reales es mucho más importante que la experiencia con los métodos ágiles. Se estima que entre el 25% y 33% del personal del proyecto debe ser competente y con experiencia, pero la necesaria incluso podría ser tan sólo de un 10% si los equipos practican la programación en parejas [43].

Las diferentes formas de refactorización o mejora en el diseño del código existente sin cambiar la funcionalidad del sistema, facilitan la simplificación de declaraciones complejas, abstrae soluciones comunes en código reutilizable, y elimina código duplicado. Producto versus documentación del proyecto, es un punto que ha llamado mucho la atención en las discusiones sobre ágil. Si el objetivo debe ser comunicarse efectivamente, la documentación debe ser la última opción pero a veces es necesaria a fin de retener información crítica. Revisores externos no pueden detectar fácilmente situaciones que conduzcan a errores arquitectónicos debido a la falta de documentación [43].

En los procesos de desarrollo tradicionales, hay una clara separación de funciones y una tendencia a organizar a las personas en torno a pequeños equipos componentes coordinados por un director de proyecto. Estos equipos tienen menos control e influyen sobre una parte del producto, perdiendo la capacidad de cooperación y necesitando mayor sincronización entre equipos cuando se introducen nuevas funciones en la organización. En el enfoque ágil, un equipo auto organizado decide cómo se coordina el trabajo y tiene el control completo sobre el proceso de desarrollo e introducción de nuevas características [47].

Los métodos de desarrollo ágil han sido cuestionados en el ámbito profesional y académico en los siguientes aspectos [41]: *el desarrollo ágil no es nuevo, estas prácticas han estado presentes en el desarrollo de software desde la década de 1960; la falta de atención a la*

arquitectura lleva a tomar decisiones de diseño poco óptimas; exiguo apoyo científico a muchas de las afirmaciones realizadas por la comunidad ágil; las prácticas en XP raras veces son aplicables y los valores sociales que adopta impulsa a los equipos ágiles a tomar decisiones ineficaces muchas veces contrarias a lo que sus miembros desean; los métodos de desarrollo ágil son adecuados para pequeños equipos no así para proyectos más grandes.

Una encuesta en línea realizada a profesionales de desarrollo de software, provenientes de varias industrias de todo el mundo, la mayoría de TI y empresas de consultoría, identificó los factores relacionados con la aplicación de una metodología de desarrollo ágil, alguno de los cuales están completamente bajo el control de la gestión: el *tamaño de la empresa,* la *formación en una metodología* permite a una organización desarrollar conocimientos y estar mejor preparados para implementarla; el *apoyo a la gestión y la participación activa* en la implementación de la metodología mejora el éxito de cualquier proyecto empresarial; el *acceso a recursos externos* tales como consultores, libros y revistas aumentará el conocimiento de la metodología por parte del equipo de desarrollo y cómo explotar las características de esa metodología para llevar beneficios a la organización [50].

Cada año son más, las pequeñas y grandes compañías que conducen sus enfoques hacia el desarrollo ágil, tales como HP, IBM, Oracle, y Microsoft. Las razones más importantes para la adopción ágil son: *acelerar el tiempo de salida al mercado, aumentar la productividad, visibilidad y capacidad para gestionar las prioridades cambiantes* [38].

Algunos de los profesionales que participaron en la definición de los métodos ágiles, creen que éstos no desplazarán a los métodos tradicionales. Por el contrario, estiman que los métodos ágiles y tradicionales tendrán una relación beneficiosa para ambos, en la que factores como el número de personas que trabajan en un proyecto, dominio de aplicación, criticidad e innovación determinarán qué proceso seleccionar [41].

Visión actual de la adopción ágil en la Industria de Software

La comunidad de investigadores ha dirigido su atención a asuntos relacionados con el desarrollo ágil de software desde que el manifiesto ágil fue declarado en el año 2001. Integrar las prácticas de desarrollo ágil con las tradicionales como una solución posible para superar las limitaciones de cada enfoque particular, y cuestiones de adopción en los dominios específicos que no son inherentemente propicios a ágil, son algunos de los enfoques estudiados actualmente.

Tabla 5: Panorama actual del desarrollo ágil de software.

Autor/es	Enfoque de estudio	Descripción
Mishra Deepti, Mishra Alok [58]	Adaptación de la metodología ágil Extreme Programming (XP) a proyectos de software con dominios complejos.	Identifica las prácticas integradas con éxito desde el punto de vista del desarrollo de software ágil y cómo superar los riesgos y limitaciones en cada fase de desarrollo de un proyecto complejo, a partir de un caso de estudio industrial. Además proporciona un conjunto de lecciones aprendidas sobre cómo los métodos ágiles podrán ser adoptados, combinados y utilizados en este tipo de proyectos.
Górski Janusz, Łukasiewicz Katarzyna [59]	Determinación de los niveles de riesgos en la aplicación de las prácticas ágiles asociadas con Scrum y eXtreme Programming (XP) en el desarrollo de software de seguridad crítica.	Presenta un caso de estudio ficticio en el ámbito académico, con el objetivo de recoger opiniones de ingenieros de software sobre los riesgos relacionados con la integración de las prácticas ágiles en proyectos de desarrollo de software de seguridad crítica.
Peñalver Romero Gladys Marsi, García de la Puente Sergio Jesús, Meneses Abad Abel [60]	SXP, un híbrido que tiene como base las metodologías ágiles: Scrum y eXtreme Programming.	La metodología SXP desarrollada en el ámbito académico ha sido estudiada, refinada y adaptada, durante sus dos años de despliegue y ejecución. Aún le queda la profunda convicción al grupo de investigadores de la misma que es perfectible aún más, sin perder de vista su carácter ágil y revolucionador.
M Rizwan Jameel Qureshi [61]	Propone un nuevo modelo eXScrum que es una versión extendida de las metodologías Scrum y XP.	El modelo eXScrum se logra mediante la integración de las prácticas de gestión de proyectos de Scrum y de ingeniería de software de XP. El modelo propuesto es validado en un caso de estudio controlado.
Mohammed S. Seyam, Galal H. Galal-Edeen [62]	Marco "Tragile", un marco híbrido que combina las prácticas tradicionales y ágiles.	El marco "Tragile" para proyectos de desarrollo de Sistemas de Información, combina directrices y prácticas de metodologías de desarrollo ágiles y tradicionales. Se evalúa la viabilidad del marco en un caso de estudio industrial.

Autor/es	Enfoque de estudio	Descripción
Rashmi Popli, Anita, Naresh Chauhan [63]	Una función de mapeo permite transformar cualquier modelo tradicional al ambiente ágil.	Propone adoptar un modelo ágil, y describe una función de mapeo para convertir un modelo tradicional al nuevo modelo ágil, haciendo coincidir parámetros de asignación.

Metodologías más relevantes en el desarrollo ágil

"Una metodología permite determinar tareas con miras a la mejora del esfuerzo realizado por el equipo de recursos humanos involucrados. Además, la proliferación y empleo de éstas, aseguran el logro de los objetivos de un proyecto software, integrando estas técnicas y métodos dentro de un ciclo completo de realización del proyecto" [42].

La elección de la mejor metodología en la etapa inicial de un proyecto, es esencial para el cumplimiento de los objetivos del equipo, para guiar y organizar las etapas y actividades de desarrollo de software que permitan obtener un producto software de valor para el cliente.

El mundo está cambiando y la ingeniería de software también, desarrollando procesos que permiten enfrentar o abrazar el cambio. *"La industria y la tecnología mueven rápidamente los requerimientos a altas tasas de cambio que empantanan a los métodos tradicionales de desarrollo de software"* [43].

"El desarrollo ágil de software es un grupo de metodologías basadas en principios similares. Su objetivo fue esbozar los valores que deberían permitir a los equipos de trabajo desarrollar software rápidamente y responder a los cambios que puedan surgir a lo largo del proyecto. Se pretendió ofrecer una alternativa a los procesos de desarrollo tradicionales, caracterizados por ser rígidos y dirigidos por la documentación generada en cada una de las actividades abordadas. Estas metodologías promueven: un proceso de gestión de proyectos que fomenta el trabajo en equipo, organización y responsabilidad propia, un conjunto de mejores prácticas de ingeniería que permiten la entrega rápida de software de alta calidad, y un enfoque de negocio que alinea el desarrollo con las necesidades del cliente y los objetivos de la organización" [42].

El enfoque ágil promueve ciclos cortos de desarrollo en forma de pequeños proyectos que aseguran el cumplimiento de las entregas incrementales. Los múltiples ciclos activan en el equipo de desarrollo la adaptabilidad al cambio y la oportunidad para abordar requerimientos emergentes [38].

"Ser ágil es tener el valor de cambiar" [46]. Los métodos ágiles tales como Scrum y Programación Extrema (XP), funcionan bien para proyectos dentro de contextos particulares: pequeños equipos que comparten el espacio de trabajo; clientes que pueden tomar decisiones sobre los requerimientos; requerimientos que cambian durante semanas o meses; contratos de precio o alcance variable; y pocas restricciones legales o reglamentarias sobre los procesos de desarrollo. Los valores y principios son abstractos, las prácticas ágiles son concretas. "El contexto debe ser entendido antes de que los métodos puedan ser elegidos o las prácticas aplicadas" [46]. Un estudio Grounded Theory corroborada por cuatro casos de estudios, distingue las prácticas que son adheridas sin cambios por el equipo de desarrollo como independientes del contexto, de las dependientes que evolucionan para adaptarlas al contexto de sus proyectos: offshore o desarrollo distribuido; proyectos en el que los clientes no están disponibles; requerimientos que cambian rara vez; o la necesidad legal de certificación de procesos y productos [46].

El desarrollo ágil ha sido ampliamente adoptado por la industria de software [38], *siendo la Programación Extrema (XP) el enfoque más conocido y aceptado dentro del desarrollo de software* [43][44]. Actualmente existen diferentes metodologías ágiles de desarrollo de software, que comparten prácticas, características y enfoques comunes, pero cada una es única en su propia estrategia de implementación [47].

Tabla 6: Resumen de las principales Metodologías de desarrollo ágil.

Método Ágil	Fuente	Enfoque	Prácticas y Proceso	Características
eXtreme Programming (XP)	Encuentra sus orígenes en la comunidad Smalltalk a finales de los 80, en colaboración entre Kent Beck y Ward Cunningham, prueban formas de desarrollar software diferente a las existentes en ese entonces. Estas prácticas refinadas en los años 90, fueron aplicadas por Kent Beck trabajando de consultor en múltiples proyectos, particularmente C3 (Chrysler Comprehensive Compensation) (1993-1997), considerado el proyecto de creación de XP.	Es un enfoque disciplinado que hace hincapié en la satisfacción del cliente y permite a los desarrolladores de software responder con confianza a los cambiantes requerimientos de software, incluso tarde en el ciclo de vida [65].	Define un proceso iterativo e incremental con pruebas unitarias continuas y entregas frecuentes [66]. El proceso de desarrollo consiste de tres etapas: Interacción con el cliente, Planificación del Proyecto, y Diseño y Desarrollo de Pruebas. Algunas de las prácticas de XP son: Juego de Planificación, Refactorización, Programación por pares, Integración continua y Cliente in-situ [66].	Apropiado para proyectos con requerimientos imprecisos y cambiantes [49][53], y donde existe un alto riesgo técnico [49]. Las características de la Metodología XP son las siguientes [43]: Debido a que el equipo de desarrollo necesita ser localizado, el tamaño del equipo generalmente aceptado es de 2 a 10 personas XP tiene una longitud de iteración de dos semanas. Los equipos distribuidos no son compatibles. La mayoría coincide en que no hay nada en XP que deba limitar su aplicabilidad para sistemas críticos.
Scrum	Ken Schwaber y Jeff Sutherland presentaron de forma conjunta y por primera vez Scrum, en 1995, como resultado del aprendizaje de la aplicación de prácticas de desarrollo de software a lo largo de los años anteriores.	El marco de trabajo de procesos emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control de riesgo [51]. Consiste en la aplicación de las ideas de la teoría de control de procesos industriales al desarrollo de software que resulta en un enfoque que reintroduce las	La metodología Scrum [66]: Respeta el ciclo de vida evolutivo y la entrega incremental, el ciclo de desarrollo se divide en una serie de iteraciones (Sprint). Propone tres fases: Planeamiento, Desarrollo y Finalización, y las siguientes prácticas: Reunión de Planeamiento del Sprint, Sprint, Reuniones Diarias, Revisión del	Indicada para proyectos con alta tasa de cambio de requerimientos y para gestionar el desarrollo de productos complejos. Las características de la Metodología Scrum son las siguientes [43]: El personal de desarrollo se divide en equipos de hasta siete personas. Un equipo completo debe incluir al menos un desarrollador, ingeniero de

Método Ágil	Fuente	Enfoque	Prácticas y Proceso	Características
		ideas de flexibilidad, adaptabilidad y productividad [65].	Sprint y Retrospectiva del Sprint.	control de calidad, y un documentalista. Se sugiere una longitud de iteración máxima de 30 días. Si bien Scrum menciona de forma explícita el apoyo a los equipos distribuidos, un proyecto puede consistir en varios equipos que podrían ser fácilmente distribuidos. Scrum no aborda explícitamente la cuestión de la criticidad del sistema.
Lean Development	Iniciado por Bob Charette, se basa en el éxito de Fabricación Lean aplicada en la industria automotriz en los años 1980 [43]. Muchos estuvieron influenciados por los métodos Lean, Mary y Tom Poppendieck. Mary, trabajó en una fábrica que usaba el método Lean, y Tom era un desarrollador de software. De ahí que Mary y Tom Poppendieck sean los pioneros en la aplicación de Lean al software. Y que escribieran el libro que ha inspirado las ideas de Lean aplicado al desarrollo software[9].	Consiste en la aplicación de Lean al proceso de desarrollo de software, se centra principalmente en las personas y la comunicación. Tiene un claro enfoque en la eliminación de los "residuos" (todo lo innecesario del que se puede prescindir), que se van generando durante el proceso de desarrollo.	Se basa en siete principios: Eliminar los desperdicios, Amplificar el aprendizaje, Decidir lo más tarde posible, Entregar lo más rápido posible, Capacitar y empoderar al equipo, Construir con integridad y Ver el todo.	Lean es aplicable a todos los ámbitos, desde el propio desarrollo hasta la misma empresa donde se produce, pasando por clientes y proveedores. Debido a que Lean Development es más una filosofía de gestión que un proceso de desarrollo, el tamaño de equipo, la longitud de la iteración, la distribución del equipo, y la criticidad del sistema no se contemplan directamente [43].
	Surgió a finales de 1990 a partir de una colaboración entre Jeff DeLuca y Peter	Se enfoca en iteraciones cortas que entregan funcionalidad tangible,	Define un proceso iterativo, con iteraciones cortas de dos semanas	Las características de la Metodología FDD son las siguientes [43]:

[9] http://www.javiergarzas.com/2012/01/lean-software-development.html

Método Ágil	Fuente	Enfoque	Prácticas y Proceso	Características
Feature Driverl Development (FDD)	Coad. DeLuca fue contratado para salvar un defecto, en un complicado sistema de préstamo. El contratista anterior había pasado dos años produciendo más de 3.500 páginas de documentación, pero no había código. DeLuca comenzó desde el principio y Coad contratado para ayudar con el modelado de objetos. Combinando sus experiencias anteriores, desarrollaron el enfoque de desarrollo FDD [43].	en cómo se realizan las fases de diseño y construcción, y en aspectos de calidad durante todo el proceso.	como máximo. El ciclo de vida consta de cinco pasos: Desarrollo de un modelo global, Construcción de una lista de funcionalidades, Planeación por funcionalidad, Diseño por funcionalidad y Construcción por funcionalidad [64]. FDD se enfoca en las fases de diseño (diseño por característica) y desarrollo (construcción por característica) siendo estas las etapas del proceso que se iteran [67].	Considera una longitud de iteración de hasta dos semanas. Está diseñado para múltiples equipos y, aunque no tiene soporte incorporado para entornos distribuidos, puede ser adaptable. FDD no se refiere específicamente a la criticidad de sistemas.
Adaptive Software Development (ASD)	Se origina de una metodología de desarrollo rápido para aplicaciones impulsada por Jim Highsmith y Bayer Sam.	ASD se enfoca en aplicar las ideas que se originaron en el mundo de los sistemas complejos, adaptación continua del proceso al trabajo. Acepta el cambio continuo como norma, enfatiza el constante aprendizaje, y la intensa colaboración entre los desarrolladores y clientes [65].	Es un proceso iterativo, tolerante a los cambios, orientado a los componentes de software más que a las tareas [64], y orientado por los riesgos [67]. Define tres etapas en un ciclo de desarrollo dinámico: Especulación, Colaboración y Aprendizaje. La revisión tiene como objetivo aprender de los errores cometidos y volver a iniciar el ciclo de desarrollo [64].	ASD fue diseñado para proyectos que se caracterizan por alta velocidad, alta tasa de cambio e incertidumbre [65].

Método Ágil	Fuente	Enfoque	Prácticas y Proceso	Características
Dynamic Systems Development Methodology (DSDM)	DSDM[10] fue creado inicialmente en 1994 con la colaboración de un gran número de practicantes de proyectos a través de empresas que estaban buscando construir un proceso de desarrollo rápido de aplicaciones (RAD) de calidad, que formaron el Consorcio DSDM como una organización sin fines de lucro para gestionar el intercambio, la explotación y evolución del marco DSDM.	El DSDM en lugar de fijar la cantidad de funcionalidad en un producto, y luego el tiempo y los recursos de ajuste para llegar a esa funcionalidad, asigna el tiempo y los recursos y, a continuación, ajusta la cantidad de funcionalidad en consecuencia [65].	Proceso iterativo e incremental, compuesto de cinco etapas de desarrollo: Estudio de Viabilidad, Estudio del Negocio, Modelado Funcional, Diseño y Construcción, e Implementación. La iteración se produce en las tres últimas etapas, y prevé retro-alimentación en todas [66].	DSDM es igualmente eficaz en las pequeñas y sencillas soluciones o grandes y complejos proyectos corporativos. DSDM se ha utilizado con eficacia en soluciones TI y no TI, y no se trata sólo de desarrollo de software. Debido a la naturaleza del marco DSDM, no especifica el tamaño de equipo, longitudes exactas de iteración, distribución, o la criticidad del sistema [43].
Crystal Methodologies	Los métodos de Crystal fueron desarrollados por Alistair Cockburn a principios de 1990, para enfrentar la pobre comunicación en el desarrollo del producto [43].	La filosofía de Crystal se enfoca en el desarrollo como un juego cooperativo de invención y comunicación cuya meta principal es entregar software útil, limitado por los recursos a utilizar [67]. Enfatiza en esfuerzos por mejorar las habilidades de los integrantes del equipo y definir políticas de trabajo [64].	La familia de metodologías Crystal [67]: Se basa en los conceptos de Rational Unified Process [RUP] y está compuesta por Crystal Clear, Crystal Yellow, Crystal Orange y Crystal Red, el nivel de opacidad del color en el nombre indica un mayor número de personas implicadas en el desarrollo, un mayor tamaño del proyecto y, por lo tanto, la necesidad de mayor control en el	Las Metodologías Crystal [43]: Se han construido en apoyo a equipos distribuidos. Se adaptan a cualquier tamaño de equipo. La longitud de iteración es de hasta 4 meses para proyectos grandes y altamente críticos.

[10] http://www.dsdm.org/

Método Ágil	Fuente	Enfoque	Prácticas y Proceso	Características
			proceso. Los ciclos donde se crean los incrementos no deben exceder cuatro meses y es necesario realizar un taller de reflexión después de cada entrega para afinar la metodología.	

A continuación se describen las metodologías Scrum, Programación Extrema (XP) y Lean Software Development, consideradas en el Capítulo 5.

Scrum

Ikujiro Nonaka y Hirotaka Takeuchi utilizaron el término Scrum en 1987, para denominar un nuevo tipo de proceso de desarrollo de productos iniciada en Japón, basada en una estrategia utilizada en rugby en la que todos los integrantes del equipo actúan juntos para avanzar la pelota y ganar el partido, similar al tipo de proceso que proponían: adaptable, rápido, auto organizable y con pocos descansos [52].

Ken Schwaber y Jeff Sutherland presentaron de forma conjunta y por primera vez Scrum, en la conferencia OOPSLA en 1995, como resultado del aprendizaje de la aplicación de Scrum en el desarrollo de software a lo largo de los años anteriores.

Ken Schwaber y Jeff Sutherland explican el proceso de gestión y control de Scrum, en la Guía Scrum[11] disponible de forma gratuita en *Scrum.org*[12], creado en 2009 por Ken Schwaber. Esta organización tiene su sede en Boston, Massachusetts (EE.UU.), ofrece todas las herramientas y recursos que necesitan los profesionales de Scrum y expertos en agilidad para entregar valor utilizando Scrum, facilita la transferencia de conocimiento, y define la formación líder de Scrum en la industria para los profesionales de todos los niveles.

La Guía mencionada, define *Scrum como un marco de trabajo de procesos que ha sido usado para gestionar el desarrollo de productos complejos, y emplea un enfoque iterativo e incremental para optimizar la predictibilidad y el control de riesgo* [51].

Una encuesta realizada a administradores y desarrolladores que trabajan principalmente en entornos distribuidos, pertenecientes a 23 empresas TI ubicadas en distintos países de la antigua Yugoslavia, revela que Scrum es una de las prácticas más utilizadas en las empresas encuestadas, y muestra los beneficios que las prácticas ágiles tienen sobre los costos y la puntualidad del proyecto [47].

La teoría de Scrum se fundamenta en el control de procesos basado en el conocimiento que procede de la experiencia y en la toma de decisiones sustentadas en dicho conocimiento, su

[11] http://www.scrumguides.org/

[12] https://www.scrum.org/

implementación se soporta en tres pilares: transparencia, inspección y adaptación. La transparencia requiere que los aspectos significativos del proceso sean visibles para aquellos que son responsables de los resultados, para lo cual deben compartir un entendimiento común del proceso. Los usuarios de Scrum deben inspeccionar con cierta frecuencia los artefactos de Scrum y el progreso hacia un objetivo para detectar variaciones. Si se determina que uno o más aspectos de un proceso se desvían de los límites aceptables, el proceso debe ser ajustado cuanto antes para minimizar desviaciones mayores [51].

La Guía de Scrum [51] documenta este marco de trabajo, tal como ha sido desarrollado y mantenido por más de veinte años por sus fundadores, y consiste en Equipos Scrum, eventos, artefactos y las reglas que gobiernan las relaciones e interacciones entre ellos:

Equipo Scrum (Scrum Team)

Los equipos Scrum son auto organizados y multifuncionales, entregan productos de forma iterativa e incremental, maximizando las oportunidades de obtener retroalimentación y asegurando que siempre estará disponible una versión útil y funcional del producto.

El equipo Scrum consiste en:

> *Dueño de Producto (Product Owner):* es la persona responsable de maximizar el valor del producto y del trabajo del Equipo de desarrollo, tomando decisiones sobre el contenido y priorizando la Lista del Producto (Product Backlog).
>
> *Equipo de Desarrollo (Development Team):* son los profesionales que participan en la creación y entrega del incremento del producto, que se pueda poner en producción al final del sprint. El tamaño óptimo del Equipo de Desarrollo es lo suficientemente pequeño como para permanecer ágil y lo suficientemente grande como para completar una cantidad significativa de trabajo.
>
> *Scrum Master:* es un líder que está al servicio del Equipo Scrum, responsable de asegurar que Scrum es entendido y adoptado, y que el Equipo Scrum trabaja ajustándose a la teoría, prácticas y reglas de Scrum.

Eventos del Sprint

El Sprint es un bloque de tiempo (time-boxes) de un mes o menos durante el cual se crea un incremento de producto utilizable y potencialmente desplegable. Scrum establece cuatro eventos formales contenidos dentro del Sprint para la inspección y adaptación, la falta de algunos de estos eventos ocasiona una disminución de la transparencia. Dichos eventos son:

Reunión de Planificación de Sprint (Sprint Planning Meeting):

El trabajo a realizar durante el Sprint se planifica en una reunión con el esfuerzo colaborativo del Equipo Scrum.

La Reunión de Planificación de Sprint responde a las siguientes preguntas:

- *¿Qué puede entregarse en el Incremento resultante del Sprint que comienza?*
- *¿Cómo se conseguirá hacer el trabajo necesario para entregar el Incremento?*

El Dueño de Producto discute el objetivo que el Sprint debería lograr y los Elementos de la Lista de Producto que, si se completan en el Sprint lograrían el Objetivo del Sprint (*Sprint Goal*). El Equipo de Desarrollo trabaja para proyectar la funcionalidad que se desarrollará durante el Sprint para formar el incremento de producto. Los elementos de la Lista de Producto seleccionados para el Sprint, más el plan para terminarlos, recibe el nombre de Lista de Pendientes del Sprint (*Sprint Backlog*).

Scrum Diario (Daily Scrum):

El Scrum Diario es una reunión de inspección y adaptación, que se realiza todos los días a la misma hora y en el mismo lugar y dirigida por el Equipo de Desarrollo, tiene una duración de 15 minutos para sincronizar sus actividades y crear un plan para las siguientes 24 horas. Esto se lleva a cabo inspeccionando el trabajo realizado desde el último Scrum Diario y haciendo una proyección sobre el trabajo que podría completarse antes del siguiente. Durante la reunión, cada miembro del Equipo de Desarrollo explica:

- *¿Qué hice ayer que ayudó al Equipo de Desarrollo a lograr el Objetivo del Sprint?*
- *¿Qué haré hoy para ayudar al Equipo de Desarrollo a lograr el Objetivo del Sprint?*
- *¿Veo algún impedimento que evite que el Equipo de Desarrollo o yo logremos el Objetivo del Sprint?*

Revisión de Sprint (Sprint Review):

Es una reunión informal que se lleva a cabo al final del Sprint para inspeccionar el incremento a los efectos de facilitar la retroalimentación de información, fomentar la colaboración y adaptar la Lista de Producto si fuese necesario.

Los asistentes determinan lo que podría hacerse para optimizar el valor, revisar la línea de tiempo, presupuesto, capacidades potenciales y el mercado para la próxima entrega prevista del producto, definiendo y ajustando los elementos de la Lista de Producto para el siguiente Sprint.

Retrospectiva de Sprint (Sprint Retrospective):

La Retrospectiva de Sprint es una reunión que tiene lugar después de la Revisión de Sprint y antes de la siguiente Reunión de Planificación de Sprint, como una oportunidad para el Equipo Scrum de inspeccionarse a sí mismo y crear un plan de mejoras que sean abordadas durante el siguiente Sprint.

El propósito de la Retrospectiva de Sprint es:

- *Inspeccionar cómo fue el último Sprint en cuanto a personas, relaciones, procesos y herramientas;*
- *Identificar y ordenar los elementos más importantes que salieron bien y las posibles mejoras; y,*
- *Crear un plan para implementar las mejoras de la forma en la que el Equipo Scrum desempeña su trabajo.*

Artefactos de Scrum

Los artefactos definidos por Scrum están diseñados para maximizar la transparencia de la información necesaria para asegurar que todos tengan el mismo entendimiento del artefacto, y son una oportunidad para la inspección y adaptación.

Lista de Producto (Product Backlog): La Lista de Producto es una lista ordenada de requerimientos del producto, los cuales tienen como atributos la descripción, la ordenación, la estimación y el valor. El Dueño de Producto (Product Owner) es el responsable de la Lista de Producto, incluyendo su contenido, disponibilidad y ordenación.

Lista de Pendientes del Sprint (Sprint Backlog): La Lista de Pendientes del Sprint es el conjunto de elementos seleccionados de la Lista de Producto, que el Equipo de Desarrollo planea llevar a cabo para la entrega del incremento y alcanzar el Objetivo del Sprint. La Lista de Pendientes pertenece únicamente al Equipo de Desarrollo y sólo éste, puede cambiarlo durante un Sprint.

Incremento: El incremento es la suma de todos los elementos de la Lista de Producto terminados durante un Sprint y el valor de los incrementos de funcionalidad de producto de todos los Sprints anteriores.

eXtreme Programming (XP)

La programación extrema o eXtreme Programming (XP), encuentra sus orígenes en la comunidad Smalltalk a finales de los 80, en colaboración entre Kent Beck y Ward Cunningham, prueban formas de desarrollar software diferente a las existentes en ese entonces. Estas prácticas refinadas en los años 90, fueron aplicadas por Kent Beck trabajando de consultor en múltiples proyectos, particularmente C3 (Chrysler Comprehensive Compensation) (1993-1997), considerado el proyecto de creación de XP.

Kent Beck sostiene [54]: *"Todo en el software cambia. Los requisitos cambian. El diseño cambia. El negocio cambia. La tecnología cambia. El equipo cambia. Los miembros del equipo cambian. El problema no es el cambio en sí mismo, puesto que sabemos que el cambio va a suceder; el problema es la incapacidad de adaptarnos a dicho cambio cuando éste tiene lugar"*, y lleva a niveles extremos un conjunto de técnicas y principios de sentido común, acuñando el nombre de Extreme Programming en 1997.

"Extreme Programming mejora un proyecto de software en cinco aspectos esenciales; la comunicación, la sencillez, la retroalimentación, el respeto y el valor. Los programadores extremos se comunican constantemente con sus clientes. Mantienen su diseño simple y limpio. Reciben retroalimentación probando su software a partir del primer día. Entregan el sistema a los clientes tan pronto como sea posible e implementan cambios como se sugiere. Cada pequeño éxito profundiza su respeto por las contribuciones únicas de cada uno y cada miembro del equipo. Con esta base Extreme programadores son capaces de responder con valentía a las cambiantes necesidades y la tecnología[13]".

La metodología de desarrollo XP propone: historias de usuario y prácticas de desarrollo, que se exponen brevemente a continuación.

Historias de usuario

Las historias de usuario son una técnica de XP utilizada para describir los requerimientos funcionales o no funcionales del software por parte del cliente, de manera que sean lo suficientemente comprensibles y delimitadas para ser descompuestas en tareas de

[13] http://www.extremeprogramming.org/

programación (*task card*) y asignadas a los programadores para ser implementadas durante una iteración (de una a tres semanas). Las historias de usuario son tarjetas de papel, cuyo tratamiento es dinámico y flexible, pueden sustituirse por otras, añadirse nuevas o ser modificadas.

Se pueden crear o modificar historias de usuarios en cualquier momento excepto cuando forman parte de una iteración en curso [67].

Prácticas

En 2004 se publicó una versión revisada de XP en la que se modifican los principios y las prácticas. Las nuevas prácticas se subdividen en dos categorías: prácticas primarias y prácticas secundarias. Esta versión, sin embargo, no ha tenido gran acogida y la mayoría de los autores siguen tomando como referencia la versión original [67].

A continuación se mencionan las *prácticas XP, originariamente propuestas por su autor*:

Planning Game: El cliente define el alcance, prioridad de las funcionalidades, cuales se deberían incluir en cada versión y la fecha de entrega de las mismas. Los desarrolladores estiman el costo y duración para la implementación de las funcionalidades, organizan el trabajo y elaboran la planificación detallada dentro de cada versión.

Versiones pequeñas: Ciclos cortos de desarrollo (*iteraciones*), facilitan entregas funcionales en versiones, beneficiando la retroalimentación entre el cliente y el equipo técnico.

Diseño simple: Consiste en diseñar en cada momento, los desarrolladores deben preocuparse únicamente por las necesidades presentes planeadas para la iteración en curso.

Testing: "Si no hacemos tests, no estaremos haciendo XP" [54]. Las pruebas guían el desarrollo del producto, las pruebas de aceptación son las primeras en realizarse y en base a ellas, se desarrolla el código correspondiente.

Refactoring: Ayuda a mantener el código simple, sin añadir o modificar la funcionalidad del sistema. Esto se corrobora ejecutando las pruebas unitarias luego de cada cambio realizado.

Integración contínua: Es un proceso que puede suceder, incluso varias veces al día, cuando una pareja de programadores integra al sistema, una tarea que haya sido probada unitariamente. Si añadida la misma, supera todas las pruebas y el sistema sigue funcionando

correctamente, se da por finalizada la integración. En caso contrario, serán los responsables de dejar el sistema de nuevo funcionando con las pruebas en su totalidad.

Propiedad colectiva del código: No hay una persona propietaria del código, cualquier programador en cualquier momento, puede hacer modificaciones en el código (*refactoring*) que agregue valor al sistema.

Programación en parejas: El código es desarrollado en parejas, cuyos roles se intercambian asegurando el conocimiento en todo el equipo de programadores, y fortaleciendo los principios de diseño simple, calidad y propiedad colectiva del código.

40 Horas semanales: Consiste en lograr un ritmo de trabajo del equipo que sea adecuado y factible de terminar la iteración o la entrega con calidad y sin trabajar horas extras.

Cliente en el sitio: Se refiere a un cliente real trabajando a tiempo completo con el equipo de desarrollo, para responder consultas de los programadores en la implementación de las historias de usuario.

Estándares de codificación: Son reglas que se definen de tal forma que el código sirva de documento. La programación en pareja y la propiedad colectiva del código, no podrían lograrse sin una codificación basada en estándares.

Lean Software Development

La filosofía *Lean*, inspirada en el éxito del proceso industrial *Lean Manufacturing* en el sector automotriz en la década de 1980, fue adaptada por Tom y Mary Poppendieck en el ámbito del desarrollo de software, y se fundamenta en los siguientes *principios*:

Eliminar el desperdicio: Consiste en desechar todo aquello que no aporta valor al cliente: retrasos en el proceso de desarrollo, comunicación lenta, requerimientos poco claros, funcionalidades y código que no son necesarios.

Amplificar el aprendizaje: El desarrollo de software es un proceso de aprendizaje continuo, la integración del cliente en el ambiente de desarrollo, iteraciones cortas cada una de ellas acompañada de refactorización y sus pruebas de integración, incrementa la retroalimentación con el cliente que a partir de los esfuerzos de desarrollo logra comprender sus necesidades y el equipo técnico aprende a satisfacerlas.

Decidir tan tarde como sea posible: en un ambiente de incertidumbre, el desarrollo iterativo permite postergar las decisiones que tienen un alto costo. A lo largo de las diferentes iteraciones el cliente puede adquirir mayor conocimiento para tomar decisiones y adoptar una aptitud previsora ante el cambio.

Entregar tan rápido como sea posible: Sin velocidad no se puede postergar las decisiones, ni garantizar que corresponden a los requerimientos actuales del cliente. Cuanto más corta son las iteraciones más rápido se obtendrá una retroalimentación del cliente para la siguiente iteración.

Potenciar el equipo: El jefe de proyecto debe ser un facilitador para que el equipo de desarrollo cumpla con sus metas y motivaciones, valorando a las personas que pueden hacer aportes al equipo y dotándoles del liderazgo suficiente para que tomen decisiones y ejecuten su trabajo.

Construir con integridad: se debe percibir al producto como integrado conceptual y técnicamente, es decir todos sus componentes funcionan correctamente en un balance entre: funcionalidad, usabilidad, confiabilidad y economía que satisface a los clientes, y con una estructura coherente que contribuya a su mantenimiento, adaptación y ampliación.

Visión global: Centrar la atención en uno de los elementos: plazos, costos o procesos, exige a los otros aumentar su carga para compensar dicha tendencia. Para aplicar este principio, se requiere un modelo de gestión comprometido en lograr un equilibrio.

CAPÍTULO 4: RIESGO Y DESARROLLO DE SOFTWARE

Fuentes de Riesgo y Modelos de Procesos del Software

Un *modelo de desarrollo de software* determina el orden en el que se llevan a cabo las actividades del proceso de desarrollo de software, es decir es el procedimiento que se sigue durante el proceso [5]. No existe un modelo ideal que se adapte a todos los proyectos de software. Los *modelos de procesos* más utilizados en la práctica actual de la ingeniería de software son: *cascada, evolutivo y componentes reutilizables* [12].

Es necesario hacer un esfuerzo adicional en el inicio del proyecto para aumentar sus probabilidades de éxito, a partir de la elección de un modelo de desarrollo adecuado a las características y circunstancias del proyecto. La elección de un modelo de ciclo de vida erróneo puede originar la omisión de tareas o una secuenciación inapropiada de las mismas, atentando contra la planificación y eficiencia del proyecto [5].

A partir de la recopilación y análisis de los principales modelos de desarrollo de software, se propone una taxonomía que los integra con el fin de facilitar la elección de un modelo apropiado para cada proyecto en particular. Dicha propuesta clasifica los modelos concretos más citados en la literatura en cinco clases abstractas: Cascada, Evolutivos, Minimización de Desarrollos, Híbridos y Ágiles, divididos en *Metodologías Tradicionales* (también llamadas *pesadas*) que son los que promueven la disciplina por medio de la planificación y la comunicación escrita, y *las Metodologías Ágiles*, que priorizan la interacción entre los individuos y la comunicación con el cliente [5].

Los cambios constantes en el alcance, constituyen una de las principales fuentes de riesgo en un proyecto, visto como cambios en los requerimientos del cliente que aturden al equipo de desarrollo durante el ciclo de vida del proyecto. Tratar el cambio es la clave para reducir los riesgos en un proyecto de desarrollo de software [17]. *La necesidad de un proyecto se genera a partir de requerimientos, la naturaleza de éstos está estrechamente relacionada con la elección del modelo de desarrollo* [5]:

- En el *modelo en Cascada* los requerimientos deben estar bien definidos desde el inicio del proyecto y la probabilidad de que cambien debe ser mínima. Se recomienda el uso de este modelo cuando los requerimientos están fuertemente acoplados o cuando son complejos, es decir cuando no es sencillo separar los requerimientos para desarrollarlos uno por uno, ya que se corre el riesgo de que el desarrollo de unos no sea compatible con el de otros.

- En el *modelo evolutivo* los requerimientos se trabajan al inicio de cada iteración para aumentarlos, corregirlos o redefinirlos. La especificación de requerimientos se desarrolla junto con el software, esto puede generar conflictos en las organizaciones donde la especificación de requerimientos es parte del contrato. Los modelos evolutivos como el *espiral* o el *incremental* tienen grandes ventajas, los clientes pueden comenzar a utilizar un sistema que tiene los requerimientos prioritarios para ponerlo a prueba y reportar sus fallas, aumentando de esta manera la probabilidad de entregar un software que opere satisfactoriamente.
- Al utilizar un *modelo de minimización de desarrollos*, se reúsa parte de los diseños modulares, cuyas funcionalidades deben ser similares a las nuevas funcionalidades que se requieren, de tal forma que el esfuerzo en modificar los componentes base no sea tan alto que el proceso se entorpezca, en lugar de agilizarse.
- El Proceso Unificado Racional (RUP) es un *modelo híbrido* que pretende sacar las ventajas de los modelos: *cascada, evolutivos y componentes reutilizables*. Es un modelo serial en el tiempo, como el modelo en cascada. La parte evolutiva e iterativa en la que se descompone cada una de las etapas, reduce los riesgos y es hasta cierto punto flexible en los cambios de requerimientos. En general es un modelo complicado que requiere una alta capacitación del administrador del proyecto y de los miembros del equipo de desarrollo, quizás este sea el motivo por el cual no es uno de los modelos más utilizados.
- Las *metodologías ágiles,* están propuestas para afrontar el problema de los requerimientos inciertos o cambiantes, las entregas iniciales tienen el objetivo de desarrollar los requerimientos esenciales del cliente y ayudan a comprender mejor el problema a solucionar. La entrega continua de nuevas versiones permite hacer frente a los cambios y a los nuevos requerimientos que emergen.

En la adopción de un modelo de desarrollo se debe tomar en cuenta la cantidad de personal, su capacidad y experiencia en el tipo de proyecto a desarrollar, e independientemente del modelo que se elija no se debe descuidar la calidad del proyecto en sus etapas iniciales, ya que esto implicaría desperdiciar el esfuerzo al corregir los errores al final, cuando son más costosos [5].

El proceso de desarrollo como un medio para gestionar los riesgos

Administrar los riesgos por medio de un proceso de desarrollo de software predecible, permite definir en forma estructurada, operacional y organizacional, una serie de actividades para gestionar los riesgos de los proyectos a lo largo de todas las fases de su ciclo de vida de desarrollo de software. En la mayor parte de los casos, esto se traduce en la creación de planes tendientes a impedir que los riesgos se transformen en problemas, o a minimizar su probabilidad de ocurrencia o impacto [27].

Decidir la metodología a seguir depende del entorno de desarrollo, el tipo de proyecto, el equipo de desarrollo y los potenciales riesgos. Dicha elección está condicionada por el nivel de riesgo del proyecto y el grado en que cada modelo de proceso apoya la gestión de riesgos. Un estudio sostiene que la mejor manera de gestionar los riesgos en los proyectos de software es, seleccionar la metodología más adecuada al proyecto y cuyo proceso de desarrollo sea un medio para gestionar los riesgos [9].

Comprender el comportamiento de los riesgos en los distintos ciclos de vida de desarrollo del software, contribuirá a tomar la decisión correcta en la elección del modelo de proceso y gestión de riesgos [17].

Modelo Cascada

Cada etapa del modelo cascada se debe completar antes de continuar con la fase posterior, y no permite la superposición entre ellas. Las etapas del modelo cascada son relativamente largas, es difícil estimar el tiempo, el costo y otros recursos necesarios para terminar con éxito cada una de ellas y los miembros del equipo, responsables de etapas específicas pasan la mayor parte de su tiempo esperando que se completen las otras etapas para poder comenzar con su trabajo. La validación del producto se limita a la fase de prueba última y la de mayor riesgo en el proceso de desarrollo, puesto que descubrir errores y riesgos en esta instancia implica un re trabajo que consume tiempo, costo y esfuerzo [9]. El modelo cascada por lo general, transmite una falsa ilusión de que todo está bien. Sin embargo, el desarrollo de software es naturalmente un proceso impredecible y como resultado de la retroalimentación con los usuarios, los cambios pueden surgir durante el ciclo de vida del proyecto. El principal inconveniente con los procesos del modelo cascada, es que se basa en la suposición de que el progreso de un proyecto de software se puede predecir con exactitud razonable desde el principio y establecer una fecha fiable de terminación. La presunción predecible del ciclo de vida garantiza un alto grado de riesgo dentro de un proyecto [17].

Desarrollo Incremental

El desarrollo incremental constituye una variante del modelo cascada, que consiste en una serie de ciclos de vida del mismo, el esfuerzo de desarrollo de software se divide entre varios incrementos de manera que los riesgos se distribuyen en múltiples iteraciones en lugar de una sola como en el desarrollo en cascada [9]. Aún así se pueden presentar algunas zonas de riesgo que afectan su aplicación. Este modelo al ser más flexible con los cambios de requerimientos tiene menos posibilidades de encontrarse con cambios de alcance, sin embargo los desafíos pueden ocurrir en cuestión de arquitectura del sistema porque no todos los requerimientos se reunieron en una sola vez para el ciclo completo [17]. Los desarrolladores tienden a posponer los requerimientos de manera que se incluyan en incrementos posteriores, pudiendo ser requerimientos principales de los cuales depende la aceptación del sistema por parte de los usuarios. Por un deficiente proceso de prueba y mantenimiento llevado a cabo al final de cada incremento, los errores que no se descubran en él, se propagan a través de incrementos posteriores y podrían resultar mucho más difícil o incluso imposible solucionarlos cuando el producto software está más avanzado. La estimación inadecuada del tiempo, costo y otros recursos necesarios para cada incremento afecta el desarrollo del proyecto, la demora en cada incremento retrasa los siguientes incrementos y la carga de tiempo adicional descansa sobre los hombros de los desarrolladores o incluso se ignoran algunos requerimientos [9].

Modelo V

Es una variante del modelo cascada que surge para hacer frente a los riesgos, por ser un proceso de desarrollo de software enfocado en las pruebas, equilibra la importancia entre el desarrollo y las pruebas [17]. Los planes de prueba se desarrollan antes de que comience el desarrollo, después del cual todos los planes de prueba se aplican y comparan sus respectivas funcionalidades con los requerimientos de la fase de recopilación. El proceso de prueba comienza temprano en el proceso de desarrollo, la planificación de las pruebas llevada a cabo en cada etapa contribuye a la identificación anticipada de los riesgos específicos del proyecto y a reducirlos a través de una gestión de procesos de mejora. Aunque el modelo V es un modelo de proceso estructurado y disciplinado, los desarrolladores lo perciben como un modelo de proceso demasiado rígido debido a la falta de flexibilidad que exhibe en contra de la naturaleza evolutiva actual de los proyectos de software [9]. El modelo V funciona bien para los proyectos de tamaño medio en el que los requerimientos estén completos y sean fáciles de entender [17].

Modelo Espiral

El modelo espiral se focaliza en el análisis de riesgos, es adecuado para grandes proyectos y de misión crítica, pudiendo llegar a ser un modelo costoso de usar [17]. Tiene como objetivo identificar y evaluar los riesgos de los proyectos de software, ayuda a reducir estos riesgos y controlar los costos del proyecto en favor de un desarrollo de software mejor controlado. Las tareas de identificación, análisis y resolución de riesgos, confían en la experiencia y habilidades de los desarrolladores en el reconocimiento y gestión de riesgos, de manera que los principales riesgos podrían permanecer ocultos en varios ciclos de vida y descubrírselos más tarde cuando se convierten en problemas reales y el costo de re trabajo para recuperarse de esos riesgos se vuelve muy alto. Programas de riesgos podrían aumentar el uso de su función iterativa, especialmente en proyectos de bajo riesgo, en el que evaluación del riesgo no está obligado a estar en este nivel de granularidad [9].

Desarrollo Ágil

La agilidad es un modelo adaptativo que exhibe una respuesta flexible al cambio, debido a su enfoque iterativo e incremental. El desarrollo ágil se centra en la comunicación entre los diferentes actores del sistema, incluidos los desarrolladores y el cliente, como fuente de identificación y planificación de riesgos, retroalimentación y cambios de requerimientos; sin embargo carece de sugerencias detalladas para la gestión de riesgos. La gestión de riesgo inherente al desarrollo ágil no es suficiente para grandes y complejos sistemas de software, los incrementos resultantes son relativamente grandes, aumentando el tiempo entre ellos, y por lo tanto requieren un mayor costo para hacer frente a cambios y errores si se descubren. Además, la gestión de comunicación es mucho más difícil en grandes equipos de desarrollo [9].

En el modelo ágil existe una alta dependencia del factor humano, en experiencia y habilidades para la comunicación efectiva con los clientes. Un representante inadecuado del cliente y/o su falta de disponibilidad influye tanto en los miembros del equipo como en el proceso de desarrollo. El enfoque ágil no es adecuado para proyectos de desarrollo de software en entornos distribuidos, requiere una comunicación cara a cara en la interacción entre el equipo de desarrollo [9].

CAPÍTULO 5: GESTIÓN DE RIESGOS EN EL DESARROLLO ÁGIL DE SOFTWARE

Enfoques que ofrecen las metodologías ágiles para la gestión de riesgos

Las metodologías aportan disciplina durante el proceso de desarrollo de software. Las metodologías ágiles ofrecen un enfoque de adaptabilidad como un medio para controlar la imprevisibilidad y un enfoque incremental, que permite que los entornos de desarrollo puedan ser mejor controlados en cada iteración y los riesgos identificados de manera oportuna.

A diferencia de otras, la ingeniería de software maneja componentes intangibles, que como tal, hace suponer que son fácilmente modificables. Para los modelos ágiles, las variaciones en la planificación de tiempo, presupuesto y calidad según los requerimientos propios del negocio, no significan necesariamente un problema para el proyecto, contrariamente a lo que sucede con las metodologías predictivas. En las metodologías ágiles los cambios no son considerados como riesgos altos en la mayoría de los casos, sino como oportunidades para retroalimentar el proyecto y obtener mayores beneficios. En tal sentido, las posibles variaciones en los requerimientos del cliente son tomadas como mejoras que aportarán un valor significativo al producto final [68].

Los riesgos están presentes en todos los proyectos de software e influyen en la selección de la metodología de desarrollo. Como se vio en el Capítulo anterior, es mejor implementar un proceso de desarrollo que ayude a una gestión controlada de los riesgos.

Existen dos puntos de vistas opuestos con respecto a la gestión de riesgos en el contexto ágil, quienes sostienen que ágil es un enfoque impulsado por el riesgo que apoya implícitamente la gestión de riesgos sin ser necesaria mejorar dicha gestión en los proyectos; y quienes creen que la naturaleza inherente de la gestión de riesgos que conduce ágil es insuficiente en algunos aspectos y debe ser mejorada [9]. Lo cierto es que, no existe un consenso definitivo sobre la necesidad de la gestión de riesgos dentro de la comunidad ágil [8].

Una investigación del estado de la gestión de riesgos en los principales modelos de procesos de software, encuentra necesaria la gestión de riesgos en todas las metodologías de desarrollo, incluso las conducidas por el riesgo [9].

Estado del arte de la gestión de riesgos en metodologías ágiles

Las metodologías ágiles han ido ganando la atención de la comunidad de software desde el año 2001, en que se constituyó la alianza de metodologías ágiles de desarrollo de software materializada en el Manifiesto Ágil. Por otra parte, un gran número de modelos de procesos se han propuestos en los últimos años para hacer frente a la necesidad de una gestión más eficaz de riesgos en proyectos de software.

Se presenta a continuación el estado actual de la *Gestión de Riesgos* desde el enfoque de las *Metodologías Ágiles* de desarrollo de software, y se identifican los métodos y herramientas utilizados en dichos dominios, como resultado de la revisión de las experiencias en el ámbito académico y de la industria de software, reportadas en revistas científicas de la especialidad desde el año 2001 a la actualidad:

Tabla 7: Estado actual de la gestión de riesgos en metodologías ágiles.

N°	Autor/es País	Modelo de Gestión de Riesgos	Metodologías ágiles	Técnicas y herramientas	Roles	Objetivo del Estudio	Ámbito de Estudio	Propuesta
1	**Lilian Arroba Medina Elizabeth [68]** Ecuador	Continuous Risk Management (CRM), SEI Gestión de riesgos basado en la realización de objetivos (Williams Ray)	Scrum	Documento de identificación de riesgos genéricos Documento de identificación de riesgos específicos Documento para priorización de riesgos Documento para gestión de riesgos Documento para control de solución de riesgos Documento para el análisis de gestión de riesgos Documento para retrospectiva de riesgos	El responsable de la gestión de riesgos, (Propietario del Producto, Scrum Master), se determina en función de las necesidades que demanden los riesgos en el proyecto, y de la disponibilidad y apertura de los miembros del grupo de trabajo.	Aplicar el proceso que plantea la metodología Scrum para minimizar los riesgos en un proyecto de desarrollo de software.	Industria	El proceso de gestión de riesgos (**Anexo 1**) se compone de cinco etapas: 1) Identificación de riesgos 2) Análisis de riesgos 3) Planteamiento y gestión de soluciones 4) Análisis de resultados 5) Continuidad de la gestión de riesgos
2	**Jaana Nyfjord, Mira Kajko-Mattsson [69]** Suecia	IEEE 1540 estándar PMBOK Método de evaluación de software (SRE)	eXtreme Programming (XP) Lean Software Development Scrum			Analizar los procesos ágiles desde una perspectiva de la gestión de riesgos, para identificar los puntos en común entre estos dos modelos.	Académico	Comparación y evaluación de los modelos de gestión de riesgos y ágiles (**Anexo 2**), utilizando una hoja de ruta que abarca los principales aspectos de la gestión de riesgos (definición de riesgo, Templates, repositorio y experiencia de base, Stakeholders, evaluación de riesgos, ciclo de vida, ambiente, organización, medidas, producto).

N°	Autor/es País	Modelo de Gestión de Riesgos	Metodologías ágiles	Técnicas y herramientas	Roles
3	Christopher R. Nelson, Gil Taran, Lucia de Lascurain Hinojosa [70] Estados Unidos de América (USA)	Continuous Risk Management (CRM), SEI	Scrum	Técnica Umbral de éxito (ToS) Técnica de multi-votación Herramienta SRE del SEI (versión para pequeños equipos)	
4	Rita Cunha, Carla Sofia Pereira, José Ângelo Pinto [71] Portugal	Gestión de riesgos según Barry Boehm PMBOK Guide Continuous Risk Management (CRM) del SEI	Scrum		

	Objetivo del Estudio	Ámbito de Estudio	Propuesta
	Discutir qué técnicas actuales para la gestión de riesgos en procesos ágiles no son suficientes y cómo los procesos pueden beneficiarse de técnicas más explícitas.	Industria	Integración de las actividades explícitas de gestión de riesgos en las tareas de procesos ágiles de Scrum, dando lugar a un conjunto de lecciones aprendidas (**Anexo 3**). *Las actividades de la gestión continua* de riesgo (identificación, análisis, planificación, seguimiento y control) se pueden integrar a las ya existentes en los procesos ágiles. Las actividades se adaptan de forma natural en las revisiones de iteración, la planificación de la iteración, y tareas de iteración de Scrum.
	Definir un modelo de gestión de riesgos para proyectos ágiles de software.	Académico	El modelo define datos y actividades (**Anexo 4**) del proceso, el cual consta de dos fases, la primera se realiza sólo una vez y la segunda se repite en cada iteración: *Paso 1:* Definición de la lista de riesgos y Planeamiento de la gestión de riesgos, a partir de la lista de riesgos de base y datos iniciales del proyecto. *Paso 2:* Plan de riesgos (incluye la definición de la lista de riesgos para la iteración y estrategias de

N°	Autor/es País	Modelo de Gestión de Riesgos	Metodologías ágiles	Técnicas y herramientas	Roles	Objetivo del Estudio	Ámbito de Estudio	Propuesta
								minimización). Monitorización y control del plan de riesgos. Al finalizar el proceso se actualiza la lista de riesgos de base con los nuevos riesgos que emergen durante el proyecto.
5	**Jaana Nyfjord, Mira Kajko-Mattsson [72]** **Jaana Nyfjord [73]** Suecia	Estándar IEEE 1540 para los procesos de ciclo de vida de software Método de Evaluación de Riesgos Software (SRE) PMBOK (PMI) AS / NZS 4360 de Gestión de Riesgos (Normas Australia, 2004).	Scrum eXtreme Programming	Prácticas workspace Informativo, historias, Desarrollo Incremental, Test-Driven Development, y programación en parejas.	Miembros del Foro de Gestión de Riesgos (RMF) Business Manager Product Manager Líder y miembros del equipo	Presentar y evaluar un modelo integrador de gestión de riesgos y procesos ágiles.	Industria	El modelo consta de dos partes: *Modelo de integración* **(Anexo 5)** proporciona orientación para la integración de la gestión de riesgos y procesos ágiles e identifica puntos de integración entre los dos procesos en una organización de software: (1) Los niveles de organización y fases de proceso, (2) Funciones y responsabilidades, (3) Canales de comunicación, y (4) Aspectos del Proceso. *Modelo integrado* **(Anexo 6)** Es un modelo de referencia contra el cual las organizaciones de software pueden comparar su proceso de gestión de riesgos. Este modelo gestiona todos los riesgos encontrados en la base de toda organización, se lo describe en función de las etapas del proceso ágil de desarrollo y los cuatro puntos del Modelo de Integración El modelo propuesto orienta para razonar acerca de la agilidad con respecto a la gestión de riesgos.

Seguidamente, se expone cada una de las propuestas de gestión de riesgos en metodologías ágiles incluidas en el estado del arte (Tabla 7):

Propuesta 1: Proceso ágil como un medio para reducir los riesgos

En los proyectos de software se presentan inconvenientes que impactan en el desarrollo del producto, para minimizar dichos riesgos un estudio [68] propone aplicar un proceso ágil que soporta implícitamente la gestión de riesgo durante todo el ciclo de vida de los proyectos:

Scrum es una metodología que permite realizar una oportuna gestión de riesgos en el proyecto, éste no es un proceso independiente de la metodología sino que se realiza de forma natural y continua en cada actividad que maneja esta metodología.

Scrum se acopla fácilmente a la gestión de riesgos, propone una revisión constante del proyecto, la detección y solución de inconvenientes de manera oportuna. Las reuniones de planificación, seguimiento y presentación de iteraciones, son herramientas que contribuyen al control de riesgos.

El proceso de gestión de riesgos se compone de cinco *Etapas*, cada una de las cuales contienen *Tareas* **(Anexo 1)**. Dicho proceso es cíclico y se debe tomar como un conjunto de actividades paralelas a las que involucran el desarrollo de la iteración, por lo que para cada iteración se debe realizar las cinco etapas que involucran la gestión de riesgos. La propuesta de reducción de riesgos basada en Scrum, se aplicó para el desarrollo e implementación de una aplicación en ambiente web de una empresa de Servicios de Telecomunicaciones de Ecuador.

La propuesta sugiere que el responsable de la gestión de riesgos, (Propietario del Producto, Scrum Master), se determine en función de las necesidades que demanden los riesgos en el proyecto, y a la disponibilidad y apertura de los miembros del grupo de trabajo. Además, recomienda formatos y documentos como ayuda para la gestión de riesgos.

Propuesta 2: Puntos en común entre procesos ágiles y gestión de riesgos

Considerando que los procesos de desarrollo ágil y de gestión de riesgos presentan diferentes enfoques, de ingeniería y de gestión, y que los procesos ágiles son conducidos por riesgos, un estudio [69] analiza éstos últimos desde una perspectiva de la gestión de riesgos, resultando que tienen muchos puntos en común entre estos dos modelos y que una combinación de ellos es posible:

El estudio se realizó en tres fases: la revisión bibilográfica, la creación de los criterios de comparación, y el análisis de los modelos de gestión de riesgos y ágiles utilizando los criterios anteriores.

Se realizó una revisión bibliográfica del estado del arte de la gestión de riesgos y su aplicación en los procesos ágiles. Se eligió un subconjunto de normas y modelos académicos y/o industriales de gestión de riesgos: estándar IEEE 1540, PMBOK y el Método de Evaluación de Software (SRE), y un subconjunto ágil: eXtreme Programming (XP), Lean Software Development y Scrum.

Posteriormente, se estudió la gestión de riesgos y se identificaron los aspectos fundamentales de la misma. Estos aspectos constituyen los siguientes criterios de comparación, que fueron utilizados para crear una hoja de ruta para la fase de comparación:

Definición del riesgo: El riesgo puede tener un efecto negativo no deseado o un efecto positivo de oportunidad. Se analiza como definen al riesgo los modelos estudiados.

Templates: La calidad de la información del riesgo en una descripción clara, completa y correcta es necesaria para la gestión eficaz del riesgo. Se estudia la cobertura de la información manejada y las sugerencias que ofrecen los modelos sobre como estructurar la información del riesgo.

Repositorio y experiencia de base: Para la gestión de análisis y seguimiento de riesgos, las organizaciones necesitan repositorios (preferiblemente electrónicos) para documentar la información del riesgo, constituir un registro de información histórica de los mismos, y ser capaces de extraer experiencias y lecciones aprendidas. Se investiga si los modelos estudiados sugieren el uso de repositorios y experiencias de base para documentar la información de la gestión de riesgos.

Stakeholders: La cobertura de las funciones de los interesados (internos o externos) en la gestión de riesgos es importante para identificar y analizar las fuentes y destinos del riesgo desde todas las perspectivas posibles. La designación de roles y responsabilidades es necesaria para la definición del proceso de gestión de riesgos. Se investiga si los modelos estudiados cubren los roles de las partes interesadas y suministran directrices sobre los mismos para la gestión de riesgos.

Evaluación de riesgos: Las taxonomías de riesgos, evaluación de atributos (probabilidad, impacto, prioridad y exposición al riesgo) y técnicas, ayudan a las

organizaciones en la planificación de diversas medidas como la designación de la gestión de riesgos, estimación del tamaño del esfuerzo de mitigación, y la identificación de las políticas que los guíen. Se examina si los modelos estudiados sugieren taxonomías, atributos y técnicas de evaluación de riesgos.

Ciclo de vida: Las actividades de gestión de riesgos pueden diferir en diversos procesos del ciclo de vida. Se estudia si los modelos de gestión de riesgos cubren todo el proceso del ciclo de vida y si ofrecen directrices para los procesos particulares.

Ambiente: Para una gestión eficaz del riesgo, el equipo de desarrollo debe considerar el proyecto en ambientes distribuidos o no distribuidos, y aspectos culturales, sociales, internacionales, políticos, etc. El grado de distribución y sus riesgos asociados deben ser considerados en la gestión de riesgos, por lo tanto se examina si los modelos estudiados consideran un entorno distribuido y si suministran directrices para abordar diversos asuntos ambientales.

Organización: Para poner en práctica una gestión efectiva de riesgos, las organizaciones deben considerar su madurez y factores como la actitud ante el riesgo, la competencia y la formación de todos los actores involucrados. En este sentido, se estudia si los modelos consideran los aspectos anteriores y proporcionan directrices para tratar con ellos.

Medidas: La medidas destinadas a eliminar los riesgos o transformarlos en aceptables, abarcan procesos y suministran continuamente información a la organización, sobre los riesgos, su progreso, recursos utilizados, la eficacia de los recursos, las políticas elegidas, y los planes de contingencia. Se analiza si los modelos sugieren procesos de gestión de riesgos y ofrecen orientación para el manejo de recursos, medición de procesos, ejecución de políticas y la integración con otros procesos.

Producto: Se debe tener en cuenta ciertos aspectos del producto (calidad del producto, su esperanza de vida, y las etapas del ciclo de vida) que pueden variar en las instancias del proceso de gestión de riesgos. Se estudia los modelos con respecto a dichas cuestiones relacionadas con el estado del producto.

Finalmente, se analizaron los modelos de gestión de riesgos y ágiles para identificar los puntos en común en ambos enfoques, utilizando los criterios de comparación mencionados.

Propuesta 3: Integración de la gestión explícita de riesgos en procesos ágiles

Los riesgos se gestionan implícitamente y de manera continua, utilizando técnicas que se encuentran en los procesos ágiles. Un estudio [70] discute qué técnicas actuales para la gestión de riesgos en procesos ágiles no son suficientes, y cómo los procesos pueden beneficiarse de técnicas más explícitas, apoyados por la experiencia de los autores en un proyecto de la industria, que utilizó Scrum:

Se comenzó con la gestión implícita de riesgos siguiendo lo prescripto por el proceso de desarrollo Scrum, el equipo percibió la necesidad de abordarlo de forma más explícita creando el *Rol de Gestor de Riesgos*, para ayudar al equipo a identificar y documentar los riesgos.

Se integraron las actividades explícitas de gestión de riesgos en las tareas ya existentes en los procesos ágiles, las actividades involucradas en la *gestión continua de riesgo* (identificación, análisis, planificación, seguimiento y control) se adaptaron de forma natural a los procesos ágiles, en las revisiones, planificaciones y tareas de las iteraciones.

Se utilizaron versiones de técnicas de identificación y evaluación de riesgos, apropiadas a pequeños equipos de trabajo con iteraciones cortas. El equipo solicitó la ayuda de expertos en gestión de riesgos de la Universidad Carnegie Mellon y el SEI, para facilitar una evaluación de riesgos de software en pequeños equipos.

Usando una técnica denominada *Umbral de éxito* (ToS), el equipo fue capaz de identificar y analizar los riesgos que podrían impedirle alcanzar el éxito del proyecto. Una lista priorizada de riesgos en forma de condición–consecuencia fue el resultado de dicha evaluación. Los riesgos fueron documentados en una wiki del equipo para permitir un fácil acceso para su visualización y frecuente actualización.

Las estrategias de mitigación de riesgos se priorizaron con otras tareas para una iteración y los planes de acción se utilizaron para realizar un seguimiento del progreso de mitigación de riesgos. Al final de cada iteración, los riesgos fueron revisados y los planes de acción sirvieron de guía al equipo en los siguientes pasos. El equipo pudo decidir si la estrategia de mitigación estaba funcionando, si el riesgo se había mitigado o convertido en un problema, y si se necesitaba cambiar dicha estrategia. Los resultados de las decisiones alimentaron la fase de planificación de la próxima iteración, nuevos riesgos identificados y nuevas tareas fueron priorizadas como parte de las estrategias de mitigación con otras tareas relacionadas con el proyecto.

Propuesta 4: Modelo de gestión de riesgos para proyectos ágiles

El uso de enfoques ágiles ha ido creciendo en los últimos diez años. En relación con los riesgos, los proyectos de software que optan por metodologías ágiles no son una excepción, es importante disponer de un modelo de orientación para la gestión de riesgos. Un estudio [71] define un *modelo de gestión de riesgos para proyectos ágiles de software*:

Como punto de partida se analizaron los modelos de gestión de riesgos. El proceso de gestión de riesgos de *Barry Boehm* subdividido en dos etapas, la evaluación y el control; el modelo de *Continuous Risk Management (CRM)* de SEI, que ofrece un ambiente propicio para la toma de decisiones, más precisamente la evaluación continua, priorización y ejecución de las estrategias de respuesta al riesgo; y el modelo definido por la *Guía PMBOK*.

El modelo define *datos y actividades* **(Anexo 4)** del proceso, el cual consta de dos fases, la primera se realiza sólo una vez y la segunda se repite en cada iteración;

Fase 1:

Partiendo de la lista de riesgos de base y de los datos iniciales del proyecto, se hace el planeamiento de la gestión para todo el proyecto incluyendo la priorización de riesgos y la asignación de responsabilidades.

Se elabora el plan de riesgos, conteniendo la lista de riesgos para la iteración y sus estrategias de minimización. Se realiza un seguimiento y control de riesgos y en caso de nuevos riesgos se actualiza el plan de riesgos.

Fase 2:

Para las siguientes iteraciones, se revisa el plan de riesgos y se hace un seguimiento y control del mismo. Al final del proceso se actualiza la lista de riesgos de base con los nuevos riesgos que emergen durante todo el proyecto.

Propuesta 5: Modelo integrador de gestión de riesgos y desarrollo ágil de software

Estudios previos muestran la necesidad de integrar prácticas explícitas de gestión de riesgos en el desarrollo ágil, a nivel de toda la organización y sin comprometer la agilidad. Se propone como solución [72] [73], integrar la gestión de riesgos y procesos ágiles, en un modelo que consta de dos partes:

(a) Modelo de integración **(Anexo 5)** que proporciona una orientación para la integración de la gestión de riesgos y los procesos ágiles, y

(b) Modelo integrado **(Anexo 6)** que sitúa a la gestión de riesgos en el proceso ágil.

El objetivo fue estudiar la gestión de riesgos en el contexto de un proceso ágil e identificar puntos de integración entre los dos procesos en una organización de software. El modelo de integración no se limitó a un modelo ágil o de gestión de riesgos particular, sino que se sintetizó a un subconjunto de modelos existentes en ambos dominios.

La *evaluación* se basó en entrevistas realizadas a diez profesionales ágiles en la industria de software sueca, con el propósito de validar la solución propuesta que incluye el Modelo de Integración y el Modelo Integrado, con respecto a los siguientes requisitos:

- El *modelo de integración* debía proporcionar a organizaciones de software una guía práctica sobre cómo integrar sus procesos de desarrollo ágiles y de gestión del riesgo.
- El *modelo integrado* debía proporcionar un modelo de referencia para que las organizaciones de software puedan examinar y comparar su gestión del riesgo.
- La solución debía proporcionar una base para discutir sobre la *agilidad con respecto a la gestión del riesgo.*

La solución se basa en el estudio de la gestión de riesgos ejecutada en el proceso ágil dentro de una organización de software. El modelo reconoce la gestión de riesgos en todo el ciclo de vida del desarrollo de software como una actividad en todos los niveles de la organización: de negocio y de ingeniería.

El modelo integrado pretende ser una referencia para comparar el sistema de gestión de riesgos, aunque de utilidad limitada por no suministrar directrices sobre cómo llevar a cabo la gestión de riesgos ágil, sólo se representa cuándo, dónde y por quién se realizó el proceso.

El Foro de Gestión de Riesgos (RMF) y los canales de comunicación ayudan a coordinar y controlar el manejo de riesgos en toda la organización, haciendo explícitos los roles responsables de la administración de los diferentes tipos de riesgos en las distintas fases del proceso y niveles de la organización; sin embargo están en conflicto con varios principios ágiles de colaboración y participación, comprometiendo así la agilidad.

Conclusiones

A continuación, se exponen las conclusiones de cada propuesta presentada:

Propuesta 1: Proceso ágil como un medio para reducir los riesgos

La aplicación de la propuesta se analizó en tres iteraciones, obteniéndose los siguientes resultados:

- La diferenciación de los riesgos en genéricos y específicos, permitió que la gestión de riesgos se realice de forma organizada y oportuna.
- El tiempo de dos semanas entre cada iteración, permitió un mejor manejo del ambiente al momento de identificar los riesgos y dar la solución respectiva.
- A medida que el desarrollo avanzó, se evidenciaron menos riesgos.
- Los riesgos identificados en cada iteración se resolvieron en su totalidad al final de cada iteración.
- Se logró que el grupo de trabajo brinde un mayor nivel de importancia a la gestión de riesgos en el proyecto de desarrollo, permitiendo que la propuesta sea implementada en las siguientes iteraciones del proyecto y en el desarrollo de otros sistemas.

La propuesta puede ser utilizada en proyectos de cualquier tamaño, siempre y cuando sean desarrollados con la metodología Scrum, y sólo maneja los riesgos que puedan ser controlados por todas las partes involucradas en el proyecto, es decir que no puede gestionar los riesgos externos a la organización (aspectos políticos, legales o de mercado).

Propuesta 2: Puntos en común entre procesos ágiles y gestión de riesgos

De la comparación y evaluación de los modelos de gestión de riesgos y ágiles, se determina que tienen muchos aspectos en común. Aunque *teórico*, el *análisis* **(Anexo 2)** muestra que existen puntos de integración entre dichos modelos. Los resultados obtenidos sólo se pueden atribuir a los modelos estudiados.

Si bien los modelos de gestión de riesgos y ágiles parecen constituir dos enfoques contrastantes, esta propuesta ayuda a identificar las partes comunes de los dos enfoques y deja en evidencia que los modelos ágiles hacen algunas afirmaciones sobre gestión de riesgos, sin suministrar sugerencias detalladas para su realización, dejando muchas áreas desatendidas.

Propuesta 3: Integración de la gestión explícita de riesgos en procesos ágiles

El ajuste a un enfoque más explícito de gestión de riesgos contribuyó al *aprendizaje de un conjunto de lecciones* **(Anexo 3)**, y tuvo ventajas significativas sobre el enfoque inicial del equipo, éste fue capaz de realizar un seguimiento del progreso de los riesgos y determinar la eficacia de las estrategias de mitigación.

Las técnicas de identificación y evaluación de riesgos utilizadas, son apropiadas para pequeños equipos de trabajo con iteraciones cortas. Las actividades involucradas en la gestión continua del riesgo se pueden integrar y adaptar de forma natural en la revisión de iteración, en la planificación de la iteración y en las tareas de iteración.

Propuesta 4: Modelo de gestión de riesgos para proyectos ágiles

Entre las principales contribuciones de este estudio, se menciona el análisis de los tres modelos de gestión de riesgos (Boehm, la Guía PMBOK, CRM del SEI), obteniéndose los siguientes resultados:

- La Guía PMBOK es el modelo que más se aproxima a enfoques ágiles, y el único que se refiere a las funciones de usuarios, permite una organización más controlada y eficiente del proceso mediante la definición de la primera fase.
- Una de las características más importantes del modelo SEI es la comunicación, que no es tan valorada por los demás modelos.

El modelo propuesto se basa en Scrum y en las ventajas de los modelos de gestión de riesgos estudiados. Es una primera propuesta del proceso, que integra las buenas prácticas de los modelos de gestión de riesgos y enfoques ágiles, que debe ser valorada en un caso práctico en un ambiente real.

Propuesta 5: Modelo integrador de gestión de riesgos y desarrollo ágil de software

Los resultados de la evaluación muestran que se cumplen los tres requisitos: el modelo de integración, el modelo integrado y el aspecto de agilidad.

La integración de la gestión de riesgos, añade más valor en contextos de desarrollo ágil con una o varias de las siguientes características: (1) equipos integrados por más de diez personas, (2) equipos distribuidos, (3) desarrollo de sistemas de seguridad críticos, sistemas complejos,

sistemas innovadores, (4) proyectos con alta exposición al riesgo, por ejemplo, dependiendo de factores tales como el entorno externo, la organización, tecnología, programación, etc.

Basándose en los resultados, los autores extraen tres conclusiones principales, (a) es una solución válida para hacer frente a la actual falta de gestión de riesgos en el desarrollo ágil, sin embargo, sólo en ciertos proyectos y organizaciones, (b) el modelo tiene que seguir desarrollándose en cuanto a la orientación que proporciona, y (c) necesita ser investigado más a fondo en cuanto a su aplicabilidad en la práctica.

CAPÍTULO 6: CONCLUSIONES Y TRABAJO FUTURO

Conclusiones

Este trabajo presenta una mirada actual de la gestión de riesgos en el desarrollo ágil de software, a partir de una revisión de la gestión de riesgos en el contexto de los procesos ágiles, y como éstos últimos pueden beneficiarse añadiendo métodos y herramientas de gestión de riesgos a los proyectos de desarrollo de software.

Los avances tecnológicos y las exigencias del mercado, dieron lugar a la aparición de nuevos métodos de gestión de procesos en los proyectos de desarrollo de software, debido a que los tradicionales no son suficientes dentro del escenario actual. Las metodologías ágiles se caracterizan por adaptarse a entornos inestables, porque incorporan mecanismos de gestión del cambio que facilitan un menor esfuerzo.

Lo cierto es que las metodologías ágiles están aportando grandes resultados en muy diversos entornos de desarrollo, cada una de ellas tiene sus fortalezas y sus debilidades **(Tabla 6)** pero no son excluyentes. No es obligatorio, ajustarse a una metodología particular a la hora de afrontar un proyecto de software; integrar las prácticas de desarrollo ágil con las tradicionales es una de las soluciones posibles para superar las limitaciones de cada enfoque particular, así como las cuestiones de adopción ágil en dominios específicos. Este trabajo contribuye con una visión actual de la práctica ágil en el desarrollo de software **(Tabla 5)**.

De las cinco producciones desarrolladas en los últimos años sobre el tema de interés **(Tabla 7)**, dos de ellas son de Suecia y fueron elaboradas por los mismos autores, las demás producciones corresponden a Portugal, Estados Unidos de América y Ecuador.

Scrum y eXtreme Programming (XP), fueron en ese orden las más utilizadas **(Tabla 7)**, siendo consistente con la afirmación de que son las metodologías ágiles más conocidas y aceptadas en la comunidad de desarrollo de software. En lo que respecta a los modelos de gestión de riesgos, el *paradigma desarrollado por SEI*, y la *guía PMBOK de PMI*, encabezan la lista de los más considerados.

El modelo desarrollado por los investigadores de Suecia, es uno de los más ambiciosos y prometedores en cuanto al propósito de integrar la gestión de riesgos en el proceso ágil, y el más completo en lo que se refiere al subconjunto de modelos de gestión de riesgos y metodologías ágiles estudiados. La solución no sólo suministra una guía para realizar la integración, sino que además es una referencia útil para posicionar y comparar la gestión de riesgos en el proceso ágil. Siendo uno de sus aportes más significativos, las evidencias teóricas

recogidas de los puntos en común identificados entre los procesos ágiles y de gestión de riesgos, utilizados para la construcción del modelo.

La integración de las prácticas y técnicas de gestión de riesgos explícitas en los procesos ágiles, puede entrar en conflicto con la filosofía ágil. En un sólo caso, se analiza si la solución se mantiene fiel al espíritu ágil, y en este sentido recomienda añadir actividades siempre que sean debidamente justificadas, de manera que no comprometan la agilidad del proceso, y al mismo tiempo mantengan y soporten las metas y prácticas de gestión de riesgos.

Los procesos ágiles ofrecen un enfoque implícito de gestión de riesgos, dejando mucho margen de mejora y áreas sin atender. Scrum evidencia una gestión eficaz de riesgos para proyectos pequeños con equipos de desarrollo de software ubicados en un mismo lugar, sin embargo los riesgos que son externos al proyecto no se pueden resolver sólo con Scrum. Todo proyecto que utilice un proceso ágil debería considerar el valor de añadir técnicas explícitas de gestión de riesgos que aumenten la probabilidad de éxito, aplicando alguno de los modelos presentados.

Si bien se puede apreciar que la convergencia de los enfoques ágiles y de gestión de riesgos en los proyectos de software está en sus inicios, y requiere un proceso de maduración y valoración a partir de un mayor apoyo empírico, es un buen punto de partida para que la comunidad ágil de software examine sus prácticas y las integre en nuevos modelos de desarrollo.

Futuras Líneas de Investigación

Una futura línea de trabajo, es comparar e identificar los aspectos comunes de los tres modelos de gestión de riesgos propuestos para el proceso Scrum y definir un único marco de desarrollo ágil de software que contenga terminología, técnicas, prácticas y estrategias comunes de implementación. Finalmente, automatizar dicho marco de desarrollo ágil a partir del desarrollo de una aplicación de software, que permita evolucionar en una base de conocimiento y mitigación de riesgos.

Para terminar, se pretende extender el marco propuesto y validarlo en entornos de desarrollo específicos no considerados hasta ahora: sistemas de seguridad crítica, equipos de desarrollo en ambientes distribuidos, sistemas innovadores, sistemas dependientes de factores externos al proyecto, etc.

REFERENCIAS BIBLIOGRÁFICAS

[1] Rodríguez José Ramón [et al.] Gestión de Proyectos Informáticos: métodos, herramientas y casos. Editorial UOC, 2007.

[2] Pressman Roger S. Ingeniería del software. Un enfoque práctico. Séptima edición. McGraw-Hill, 2010.

[3] Barry W. Boehm, TRW. Software Risk Management: Principles and Practices. IEEE Software, Vol. 8, No. 1, pp. 32-41, 1991.

[4] Amendola Luis José. Estrategias y tácticas en la dirección y gestión de proyectos. Editorial Universidad Politécnica de Valencia, 2006.

[5] Cervantes Ojeda J., Gómez Fuentes María del Carmen. Taxonomía de los modelos y metodologías de desarrollo de software más utilizadas. Universidades, vol. LXII, núm. 52, pp. 37-47, 2012.

[6] Rodríguez González Pilar. Estudio de la aplicación de Metodologías Ágiles para la evolución de Productos Software. http://oa.upm.es/1939/1/TESIS_MASTER_PILAR_RODRIGUEZ_GONZALEZ.pdf, 2008. (Acceso: 30 de Octubre de 2014)

[7] Wijewardena Thushara. Risk Management – where it fits in Scrum? http://projectized.blogspot.com.ar/2010/02/risk-management-where-it-fits-in-scrum.html (Acceso: 29 de Marzo de 2014)

[8] Thekku Veethil Satheesh. Gestión de Riesgos en Agile. http://www.scrumalliance.org/community/articles/2013/2013-may/risk-management-in-agile (Acceso: 29 de Marzo de 2014)

[9] Hijazi Haneen, Khdour Thair, Alarabeyyat Abdulsalam. A Review of Risk Management in Different Software Development Methodologies. International Journal of Computer Applications (0975 – 8887) Volume 45– No.7, 2012.

[10] Chawan P. M, Patil Jijnasa, Naik Radhika, Software Risk Management, International Journal of Advances in Engineering Sciences Vol. 3, Issue 1, January 2013.

[11] Williams Laurie. Agile Software Development Methodologies and Practices. Advances in computers, VOL. 80, 2010.

[12] Sommerville Ian. Ingeniería del software. Pearson Educación, Séptima Edición, 2005.

[13] Brooks, Frederick P. "No Silver Bullet: Essence and Accidents of Software Engineering," Computer, Vol. 20, No. 4, Abril 1987 pp. 10-19.

[14] Guide to the Project Management Body of Knowledge (PMBOK® Guide) Fifth Edition, 2012.

[15] Standish Group, "The CHAOS Manifesto", 2013.

[16] Dominguez Jorge. "The Curious Case of the CHAOS Report 2009". 1 de julio 2009. http://www.projectsmart.co.uk/the-curious-case-of-the-chaos-report-2009.php (Acceso: 05/02/2015)

[17] Verma Charu. The effects of organizational culture on risk management during software development, 2009. http://charuv.com/MSc_ITM_Thesis_Charu.pdf (Acceso: 11/05/2013)

[18] Salinas Duarte Andrés Ernesto, Obstáculos en la Gestión de Proyectos en Tecnologías de Información Y Comunicación - TICs y Posibles Soluciones, 2007. http://52.0.140.184/typo43/fileadmin/Articulos/Andres_Salinas.pdf (Acceso: 12/11/2014)

[19] Bannerman Paul L. Risk and risk management in software projects: A reassessment. The Journal of Systems and Software Volumen 81, Diciembre 2008. Pág. 2118–2133. http://www.sciencedirect.com/science/article/pii/S0164121208000897 (Acceso: 19/06/2014)

[20] Pérez Moya Osiris, Zulueta Véliz Yeleny. Proceso para gestionar riesgos en proyectos de desarrollo de software. Revista Cubana de Ciencias Informáticas Vol. 7, No. 2, Abril-Junio, 2013. Pág. 206-221 http://scielo.sld.cu/scielo.php?pid=S2227-18992013000200009&script=sci_arttext (Acceso: 06/11/2014)

[21] Pereira R. Javier, Cerpa T. Narciso, Rivas M. Mario. Factores de éxito en proyectos de desarrollo de software: Análisis de la industria chilena del software. Workshop de Ingeniería de Software, 2004. http://www.academia.edu/915974/Factores_de_%C3%A9xito_en_proyectos_de_desarrollo_de_software_an%C3%A1lisis_de_la_industria_chilena_de_software (Acceso: 05/08/2014)

[22] McLeod, L. MacDonell, S. G. Factors that affect software systems development project outcomes: A survey of research. ACM Comput. Surv. 43, 4, Article 24. 2011. http://doi.acm.org/10.1145/1978802.1978803 (Acceso: 10/05/2013)

[23] Clarke Paul, O'Connor Rory V. The situational factors that affect the software development process: Towards a comprehensive reference framework. Journal of

Information Software and Technology, Vol. 54, Issue 5, May 2012. pp. 433-447. http://dl.acm.org/citation.cfm?id=2133931 (Acceso: 18/07/2013)

[24] Kwak Y.H., Stoddard J. Project risk management: lessons learned from software development environment. Technovation 24, 2004. ww.elsevier.com/locate/technovation (Acceso: 17/07/2013)

[25] Demarco Tom, Lister Timothy. Waltzing with Bears: Managing Risk on Software Projects. 2003.

[26] Sá Silva Pedro, Trigo António, Varajão João. Collaborative Risk Management in Software Projects work-in-progress. 2012 Eighth International Conference on the Quality of Information and Communications Technology.

[27] Bertone Rodolfo, Thomas Pablo, Taquias Daniel, Pardo Sebastián. Herramienta para la Gestión de Riesgos en Proyectos de Software. XVI Congreso Argentino de Ciencias de La Computación. http://sedici.unlp.edu.ar/handle/10915/19289 (Acceso: 29/04/2015)

[28] Dhlamini John, Nhamu Isaac, Kachepa Admire. Intelligent Risk Management Tools for Software Development. SACLA. 2009 ACM. http://web.nchu.edu.tw/pweb/users/arborfish/lesson/8614.pdf (Acceso: 28/07/2014)

[29] Marichal Barrios, Pablo Daniel. Diseño conceptual de un sistema para la gestión de riesgos en proyectos de desarrollo de software. Trabajo de grado. Universidad Católica de Andrés Bello. Venezuela, 2009. http://w2.ucab.edu.ve/tesis-digitalizadas2/ths_mention/especialista-en-gerencia-de-proyectos.html?page=2 (Acceso: 16/04/2013)

[30] Ivorra Valero José. La Gerencia de Riesgos – Factor Crítico de Éxito. III Congreso Iberoamericano de Gerencia de Proyectos, Venezuela. 2002. http://webquest.carm.es/majwq/public/files/files_user/jvillalba/gerencia_del_riesgo.pdf (Acceso: 28/07/2014)

[31] Xiaosong Li, Shushi Liu, Wenjun Cai, Songjiang Feng. The Application of Risk Matrix to Software project risk management. 2009 International Forum on Information Technology and Application. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=5231375 (Acceso: 10/05/2013)

[32] Marvin J. Carr, Suresh L. Konda, Ira Monarch, F. Carol Ulrich y Clay F. Walker. Taxonomy-Based Risk Identification. Technical Report. 1993. CMU/SEI-93-TR-6. http://resources.sei.cmu.edu/library/asset-view.cfm?assetid=11847 (Acceso: 26/01/2016)

[33] Higuera Ronald P., Haimes Yacov Y. Software Risk Management. Technical Report CMU/SEI-96-TR-012. Software Engineering Institute. Carnegie Mellon University. 1996.

[34] Higuera Ronald P., Gluch David, Dorofee Audrey J., Murphy Richard L., Walker Julie A. An Introduction to Team Risk Management, 1994. http://repository.cmu.edu/sei (Acceso: 15/01/2015)

[35] CMMI® para Desarrollo, Versión 1.3. Mejora de los procesos para el desarrollo de mejores productos y servicios. Technical Report CMU/SEI-2010-TR-033. 2010.

[36] Marcos de riesgos de TI. Risk IT. Basado en COBIT® . ISACA. www.isaca.org. 2009.

[37] Montenegro Hoyos John Jairo, Rivera Restrepo Mariela Cecilia. Risk IT como complemento a la gestión de riesgos en compañías de la industria de software. 2011. http://bibliotecadigital.icesi.edu.co/biblioteca_digital/handle/10906/68005 (Acceso: 03/06/2015)

[38] Moniruzzaman A B M, Hossain Syed Akhter. Comparative Study on Agile software development methodologies. Global Journal of Computer Science and Technology (c) Volume 13 Issue 7 Version I. 2013. http://arxiv.org/ftp/arxiv/papers/1307/1307.3356.pdf (Acceso: 07/04/2014)

[39] Bustard David, Wilkie George, Greer Des. The Maturation of Agile Software Development Principles and Practice: Observations on Successive Industrial Studies in 2010 and 2012. 20th Annual IEEE International. Conference and Workshops on the Engineering of Computer Based Systems (ECBS), 2013.

[40] Torgeir Dingsøyr, Sridhar Nerur, VenuGopal Balijepally, Nils Brede Moe. A decade of agile methodologies: Towards explaining agile software development. The Journal of Systems and Software 85, 2012. http://www.sciencedirect.com/science/article/pii/S0164121212000532 (Acceso: 23/09/2014)

[41] Dybå T., Dingsøyr T. Empirical studies of agile software development: a systematic review. Information and Software Technology 50, 833–859, 2008. http://dl.acm.org/citation.cfm?id=1379989 (Acceso: 13/06/2015)

[42] Mariño Sonia I., Godoy Maria V., Alfonzo Pedro L. Propuestas y revisión de metodologías de la Ingeniería del Software. WICC 2014 XVI Workshop de Investigadores en Ciencias de la Computación, 2014. http://sedici.unlp.edu.ar/handle/10915/41409 (Acceso: 05/07/2015)

[43] Cohen David, Lindvall Mikael, Costa Patricia. An Introduction to Agile Methods. Advances in computers, vol. 62, 2004. http://www.cse.chalmers.se/~feldt/courses/agile/cohen_2004_intro_to_agile_methods.pdf (Acceso: 02/10/2015)

[44] Figueroa Roberth G., Solís Camilo J., Cabrera Armando A. Metodologías tradicionales vs. metodologías ágiles. Universidad Técnica Particular de Loja, Escuela de Ciencias en Computación.

[45] Kettunen Petri. Adopting key lessons from agile manufacturing to agile software product development. A comparative study. Technovation 29, 2009. Pág. 408–422. http://www.sciencedirect.com/science/article/pii/S0166497208001302 (Acceso: 23/09/2014)

[46] Hoda Rashina, Kruchten Philippe, Noble James, Marshall Stuart. Agility in Context. OOPSLA/SPLASH'10, October 17–21, 2010. http://dl.acm.org/citation.cfm?id=1869467 (Acceso: 23/09/2014)

[47] Stankovic Dragan, Nikolic Vesna, Djordjevic Miodrag, Cao Dac-Buu. A survey study of critical success factors in agile software projects in former Yugoslavia IT companies. The Journal of Systems and Software 86 (2013) 1663 – 1678. http://dl.acm.org/citation.cfm?id=2467408 (Acceso: 23/09/2014)

[48] Herrera Uribe Eliécer, Valencia Ayala Luz Estela. Del Manifiesto Ágil sus valores y principios. Scientia et Technica Año XIII, No 34, Mayo de 2007. Universidad Tecnológica de Pereira. http://www.redalyc.org/pdf/849/84934064.pdf (Acceso: 29/07/2015)

[49] Arteaga Ceballos Claudia Milena, Barrera Pineda Fernando Augusto, Chaparro Pedraza Jarold Javier. Factores claves para la gestión de proyectos tecnológicos. Vol. 2 ǀ No 1. TIA.

[50] Livermore Jeffrey A. Factors that Significantly Impact the Implementation of an Agile Software Development Methodology. Journal of software, Vol. 3, No. 4, Abril de 2008. http://ojs.academypublisher.com/index.php/jsw/article/view/1943 (Acceso: 04/06/2014)

[51] Schwaber Ken, Sutherland, Jeff. La Guía de Scrum. La Guía Definitiva de Scrum: Las Reglas del Juego. Julio de 2013. http://www.scrumguides.org/ (Acceso: 11/10/2015)

[52] Rodríguez González Pilar. Estudio de aplicación de metodologías ágiles para la evolución de productos software. Tesis de Máster. Máster en Tecnologías de la Información. Septiembre de 2008. http://oa.upm.es/1939/ (Acceso: 21/10/2014)

[53] Pérez Ramírez Danay, Oliveros Guntín Yoanna, Alvarez Alonso Yanniel, Coello Mena Jorge. Métodologías Ágiles ¿Cómo desarrollo utilizando XP? Convención Científica de Ingeniería

y Arquitectura. 2008. http://ccia.cujae.edu.cu/index.php/siia/siia2008/paper/viewFile/1174/246 (Acceso: 20/08/2015)

[54] Acebal César F., Cueva Lovelle Juan M. eXtreme Programming (XP): un nuevo método de desarrollo de software. NOVATICA/UPGRADE marzo.-abril. 2002 Nº 156. http://www.ati.es/novatica/2002/156/156-8.pdf (Acceso: 12/10/2015)

[55] The Institute of Electrical and Electronics Engineers, Inc. IEEE Standard for Software Life Cycle Processes—Risk Management. IEEE Std 1540-2001.

[56] Hersyah Mohammad Hafiz, Surendro Kridanto. ICT Risk Management Capability Dimension for Smart Education - A two dimensional approach. ICT for Smart Society (ICISS), 2013 International Conference.

[57] Ernawati Tati, Suhardi, Nugroho Doddi R. IT Risk Management Framework Based on ISO 31000:2009. International Conference on System Engineering and Technology, 2012. Bandung, Indonesia.

[58] Mishra Deepti, Mishra Alok. Complex software project development: agile methods adoption. Journal of Software Maintenance and Evolution: Research and Practice, 2011. http://dl.acm.org/citation.cfm?id=2228988 (Acceso: 21/05/2013)

[59] Górski Janusz, Łukasiewicz Katarzyna. Towards Agile Development of Critical Software. A. Gorbenko, A. Romanovsky, V. Kharchenko (Eds.): SERENE 2013, LNCS 8166, pp. 48–55, 2013. http://link.springer.com/chapter/10.1007/978-3-642-40894-6_4 (Acceso: 23/09/2014)

[60] Peñalver Romero Gladys Marsi, García de la Puente Sergio Jesús, Meneses Abad Abel. SXP, metodología de desarrollo de software. Serie Científica de la Universidad de las Ciencias Informáticas No. 11, Vol. 4, 2011.

[61] M Rizwan Jameel Qureshi. Empirical Evaluation of the Proposed eXSCRUM Model: Results of a Case Study IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 3, No. 2, 2011. http://arxiv.org/ftp/arxiv/papers/1202/1202.2513.pdf (Acceso: 16/06/2014)

[62] Mohammed S. Seyam, Galal H. Galal-Edeen. Traditional versus Agile: The Tragile Framework for Information Systems Development. Int. J. of Software Engineerin, IJSE Vol. 4 No. 1, 2011.

[63] Rashmi Popli, Anita, Naresh Chauhan. A Mapping Model for Transforming Traditional Software Development Methods to Agile Methodology. International Journal of Software Engineering & Applications (IJSEA), Vol.4, No.4, 2013.

[64] Mendes Calo Karla, Estevez Elsa, Fillottrani Pablo. Evaluación de Metodologías Ágiles para Desarrollo de Software. WICC 2010 - XII Workshop de Investigadores en Ciencias de la Computación, 2010.

[65] Veerapaneni Esther Jyothi, K. Nageswara Rao. Effective Implementation of Agile Practices Ingenious and Organized Theoretical Framework. (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 2, No.3, 2011.

[66] Karla Mendes Calo, Elsa Estevez, Pablo Fillottrani. Un Framework para Evaluación de Metodologías Ágiles. XV Congreso Argentino de Ciencias de la Computación, 2009.

[67] Navarro Cadavid Andrés, Fernández Martínez Juan Daniel, Morales Vélez Jonathan. A review of agile methodologies for software development. Prospect. Vol. 11, No. 2, Julio - Diciembre de 2013, págs. 30-39.

[68] Arroba Medina Lilian Elizabeth. Propuesta de aplicación de Scrum para minimizar los riesgos en un proyecto de desarrollo de software. Facultad de ingeniería de sistemas - Escuela Politécnica Nacional, 2011. http://bibdigital.epn.edu.ec/handle/15000/3760 (Acceso: 27/02/2014)

[69] Nyfjord Jaana, Kajko-Mattsson Mira. Commonalities in Risk Management and Agile Process Models. ICSEA 2007, Cap Esterel France, August 2007. http://dl.acm.org/citation.cfm?id=1306419 (Acceso: 07/01/2015)

[70] Christopher R. Nelson, Gil Taran, Lucia de Lascurain Hinojosa. Explicit Risk Management in Agile Processes. P. Abrahamsson et al. (Eds.): XP 2008, LNBIP 9, pp. 190–201, 2008. © Springer-Verlag Berlin Heidelberg 2008.

[71] Cunha, R.; Pereira, C.S.; Pinto, J.A, "Agile software project: Proposal of a model to manage risks". Information Systems and Technologies (CISTI), 2013 8th Iberian Conference on, vol., no., pp.1,5, 19-22 June 2013.

[72] Nyfjord Jaana, Kajko-Mattsson Mira. Outlining a Model Integrating Risk Management and Agile Software Development. 34th Euromicro Conference Software Engineering and Advanced Applications IEEE, 2008.

[73] Nyfjord Jaana. Towards integrating agile development and risk management. Stockholm University, 2008. http://www.diva-portal.org/smash/get/diva2:199663/FULLTEXT01.pdf (Acceso: 10/07/2014)

[74] Microsoft Solutions Framework. MSF Risk Management Discipline v.1.1 White Paper. Junio 2002. https://www.microsoft.com/en-us/download/details.aspx?id=721 (Acceso: 26/01/2016)

ANEXOS

Anexo 1: Proceso de gestión de riesgos aplicando Scrum [68].

PROPUESTA		PROCESO SCRUM	PROPUESTA	
ETAPAS	**TAREAS**	**FASES**	**DOCUMENTO PROPOUESTO**	**DESCRIPCIÓN**
Etapa 1: Identificación de riesgos	**Tarea 1:** Obtener información de riesgos genéricos.	Se debe realizar durante las primeras fases del ciclo de Scrum: generación, planificación y seguimiento del Product Backlog y Sprint Backlog.	Documento de identificación de riesgos genéricos	Propone la utilización de un documento de identificación de riesgos genéricos conteniendo la siguiente información: • Identificador del riesgo: número secuencial o nomenclatura según algún estándar. • Descripción del riesgo: detalle preciso del riesgo. • Causas. • Porcentaje de ocurrencia. • Nivel de criticidad. • Consecuencias.

PROPUESTA		PROCESO SCRUM
ETAPAS	**TAREAS**	**FASES**
	Tarea 2: Obtener información de riesgos específicos del proyecto.	
	Tarea 3: Definir grado de	Se recomienda que la tarea se

	PROPUESTA	
	DOCUMENTO PROPOUESTO	DESCRIPCIÓN
	Documento de identificación de riesgos específicos	Propone la utilización de un documento de identificación de riesgos específicos conteniendo la siguiente información: • Identificador del riesgo: número secuencial o nomenclatura según algún estándar. • Descripción del riesgo: detalle preciso del riesgo. • Causas. • Porcentaje de ocurrencia. • Nivel de criticidad. • Consecuencias.
	Documentos de identificación de	En los documentos de identificación de

PROPUESTA		PROCESO SCRUM	PROPUESTA	
ETAPAS	**TAREAS**	**FASES**	**DOCUMENTO PROPOUESTO**	**DESCRIPCIÓN**
Etapa 2: Análisis de riesgos	probabilidad de los riesgos.	realice en la fase de definición de tareas.	riesgos genéricos y específicos	riesgos genéricos y específicos, se asignan las principales causas de cada riesgo y el porcentaje de ocurrencia correspondiente al promedio de los porcentajes propuestos por cada integrante con la técnica de estimación de póker.
	Tarea 4: Establecer consecuencias de los riesgos y nivel de criticidad en el proyecto.	Se realizará en las fases de definición de responsabilidades y seguimiento de la iteración.	Documentos de identificación de riesgos genéricos y específicos	En los documentos de identificación de riesgos genéricos y específicos, se asigna el nivel de criticidad (Despreciable, Marginal, Crítico, Catastrófico) y los efectos o consecuencias asociadas al riesgo para entender el grado de impacto sobre el proyecto.
	Tarea 5: Especificar la prioridad de los riesgos.	Se desarrollará en la fase de definición de responsabilidades.	Documento para priorización de riesgos	Propone el documento para priorización de riesgos, a partir del nivel de criticidad se podrá especificar la prioridad de atención que se dará a cada riesgo (Alto, Medio, Bajo), y el orden en el cuál serán solucionados.

PROPUESTA		PROCESO SCRUM	PROPUESTA	
ETAPAS	**TAREAS**	**FASES**	**DOCUMENTO PROPOUESTO**	**DESCRIPCIÓN**
Etapa 3: Planteamiento y gestión de soluciones	**Tarea 6:** Asignar nivel de gestión de riesgos.	Se desarrollará durante la fase del ciclo de Scrum: seguimiento de la iteración.	Documento para gestión de riesgos	Se analiza la solución del riesgo siguiendo el orden de atención. Propone un documento para gestión de riesgos, donde se establece la acción a realizar: Control de crisis, Arreglar cada *error*, Mitigación de *riesgos*, Prevención, Eliminación de causas principales.
	Tarea 7: Planteamiento de soluciones.			Se asigna la solución acordada a cada uno de los riesgos, su tiempo de solución y responsable.
	Tarea 8: Ejecución de solución.			Se ejecuta la solución bajo los parámetros detallados y acordados.
	Tarea 9: Control de la solución.		Documento para control de solución de riesgos	Durante el tiempo que tome la solución se debe llevar un control de la misma, indicando el porcentaje de ejecución y el estado en que se encuentra la solución (pendiente, en curso, solucionado).
Etapa 4: Análisis de	**Tarea 10:** Realizar análisis de	Se desarrollará durante la fase del		Se asigna el número de Sprint en el que se

PROPUESTA		PROCESO SCRUM	PROPUESTA	
ETAPAS	**TAREAS**	**FASES**	**DOCUMENTO PROPOUESTO**	**DESCRIPCIÓN**
resultados	resultados acerca del grado de reducción de riesgos.	ciclo de Scrum: presentación de la iteración.	Documento análisis de gestión de riesgos Documento para retrospectiva de riesgos	analiza los riesgos y se cuantifica el porcentaje de riesgos que se solucionaron según lo planificado. Se utiliza el concepto de retrospectiva para analizar los resultados de la gestión realizada a los riesgos aún no solucionados, se determina las razones por las cuales no se solucionaron y el porcentaje ya realizado de la solución. En caso de que sea necesario se puede realizar una modificación total o parcial de la solución.
Etapa 5: Continuidad de la gestión de riesgos	**Tarea 11:** Repetir el ciclo de gestión de riesgos.			Para cada Sprint se deben realizar las cinco etapas que involucran la gestión de riesgos.

Anexo 2: Evaluación de los modelos de gestión de riesgos y procesos ágiles utilizando los criterios de comparación [69].

Criterios	Modelos de gestión de riesgos	Modelos ágiles
Definición de riesgo	Definen habitualmente el riesgo como una condición adversa o evento que afectará el proyecto. PMBOK reconoce explícitamente el impacto positivo del riesgo.	La definición ágil de riesgo es similar a la de los modelos de gestión de riesgos. XP considera el riesgo como oportunidad.
Templates	Los modelos IEEE 1540 y Método de Evaluación de Software (SRE), suministran plantillas para estructurar la información de riesgo. El Método de Evaluación de Software (SRE), ofrece una plantilla más detallada en forma de un cuestionario basado en la taxonomía para asegurar que todos los riesgos se abordan sistemáticamente.	Los modelos ágiles estudiados no sugieren plantillas específicas para la descripción de los riesgos ni proporcionan directrices, por lo que la información del riesgo debe ser gestionada por el proceso.
Repositorio y experiencia de base	Todos los modelos de gestión de riesgo estudiados sugieren la creación y el apoyo de un repositorio electrónico para documentar los riesgos y un registro de la experiencia adquirida en la mitigación de riesgos.	Los modelos ágiles no sugieren repositorios y experiencia de base, para documentar la información de la gestión de riesgos. Se usan almacenamientos electrónicos, si surge la necesidad. El espacio de trabajo informativo se utiliza en lugar de los repositorios electrónicos, la mejora de procesos y la reevaluación del riesgo, se produce continuamente en las reuniones ágiles retrospectivas.
Stakeholders	Los modelos de gestión de riesgos estudiados sugieren diferentes conjuntos de funciones de las partes interesadas. El estándar IEEE 1540 es el único que menciona a los clientes. Los modelos sólo identifican los roles que tienen responsabilidades de mitigación de riesgos conocidos comúnmente como propietarios de los riesgos y las directrices sobre la forma de gestionar los riesgos se limita sólo a los directores de proyecto.	Los modelos ágiles sugieren equipos como principales funciones de los interesados y no suministran directrices específicas para la designación de los roles y responsabilidades para la gestión de riesgos. El equipo es el dueño del proceso de gestión de riesgos, el jefe de proyecto facilita el proceso y hace visible los resultados.
Evaluación de riesgos	Todos los modelos de procesos de gestión de riesgos estudiados sugieren atributos para evaluar los riesgos basados en la probabilidad, el impacto, la gravedad y la prioridad. También sugieren diversas técnicas para la evaluación de riesgos, es decir, para asignar valores a estos atributos. El estándar IEEE 1540 establece las diferentes técnicas que se deben utilizar, mientras que el Método de Evaluación de Software (SRE), ofrece descripciones de cómo calcular la exposición del riesgo.	Los modelos ágiles no sugieren valores para los atributos del riesgo o técnicas específicas para evaluarlos. Los equipos evalúan continuamente los riesgos, a partir de una lista priorizada y asignan valores que se publican en el espacio de trabajo informativo, generalmente se muestran como notas de color que indican diferentes prioridades.
	Los procesos de gestión de riesgos	El enfoque iterativo e incremental y evolutivo

Criterios	Modelos de gestión de riesgos	Modelos ágiles
Ciclo de Vida	estudiados apoyan el ciclo de vida del producto software en un grado variable. El Método de Evaluación de Software (SRE), sólo crea una línea de base para iniciar un proceso de gestión de riesgos y no cubre ninguno de los procesos del ciclo de vida. El PMBOK es una guía para la gestión de todo tipo de proyectos, en lo que se refiere a la ingeniería de software corresponde principalmente a la fase de desarrollo. Si bien el estándar IEEE 1540, afirma que proporciona soporte para la gestión de riesgos a través de todo el ciclo de vida del software, se concluye que ninguno de los modelos estudiados cubre otros procesos del ciclo de vida, sólo se concentran en el desarrollo a nivel de proyecto.	en procesos ágiles significa que en cada iteración, las mejoras y correcciones se realizan en paralelo. Por lo tanto, el enfoque ágil cubre automáticamente la mayoría de los procesos primarios tales como el desarrollo, la evolución y el mantenimiento. Los procesos ágiles parecen cubrir mejor la mayoría de los procesos primarios, y de apoyo con las frecuentes reuniones.
Ambiente	Excepto por el Método de Evaluación de Software (SRE), los modelos de gestión de riesgos mencionan los aspectos ambientales y su importancia en la aplicación de la gestión de riesgos. Sin embargo, ninguno de ellos ofrece una guía explícita de cómo hacer frente a estos aspectos en la práctica.	Los modelos ágiles ofrecen algunas pautas para adaptar proyectos con respecto al ambiente. Los principios de XP, son algunas formas de directrices para la creación de equipos y el direccionamiento en incidencias culturales. En cuanto a la distribución, los modelos ágiles son más eficaces cuando se tratan de equipos pequeños co-localizados.
Organización	En cuanto a las cuestiones de organización, como la actitud, la madurez y la competencia, difieren en el enfoque y nivel de detalle. Ofrecen directrices generales: comunicar eficazmente los riesgos en la organización para evitar la aversión al riesgo, y herramientas de educación y formación para aumentar la madurez de la gestión de los riesgos.	Muchas de las prácticas y principios ágiles se centran en la forma de organizar los proyectos y equipos de manera efectiva con respecto a la madurez, actitud y competencia, incluyen directrices para construir relaciones, fomentar el intercambio de conocimiento y el aprendizaje en los proyectos de software.
Medidas	Todos los modelos de gestión de riesgos definen un modelo de proceso completo de gestión de riesgos y sugieren que la misma colabora con el proceso de medición para lograr el monitoreo y control efectivo. Excepto el Método de Evaluación de Software (SRE), que se centra sólo en la identificación y análisis de riesgos. Todos los modelos de gestión de riesgos estudiados proporcionan directrices sobre cómo asignar recursos a la gestión de riesgos, principalmente a funciones y responsabilidades de los recursos humanos.	Ninguno de los modelos de procesos ágiles sugiere explícitamente las fases de gestión de riesgos, sin embargo los modelos ágiles están integrados con los procesos de medición. Los modelos ágiles afirman que todos los recursos deben estar disponibles para garantizar la efectividad del proyecto, pero expresan de manera vaga el problema de asignación de recursos. Los modelos de procesos ágiles no especifican las políticas o estrategias de gestión de riesgos.

Criterios	Modelos de gestión de riesgos	Modelos ágiles
	Todos los modelos de gestión de riesgos sugieren que pueden definir políticas para el proceso de gestión de riesgos, sin embargo los modelos son muy generales. Sólo el estándar IEEE 1540 establece claramente las políticas y estrategias de gestión de riesgos.	
Producto	El estado del producto no se discute en los modelos de gestión de riesgos, excepto en el Método de Evaluación de Software (SRE), que describe los riesgos y la gestión de riesgos en relación con el estado del producto.	Los modelos de procesos ágiles no proporcionan directrices para relacionar el estado del producto a la gestión de riesgos. Cuestiones relativas al estado del producto vinculadas con la calidad y la esperanza de vida, se basan completamente en las prioridades establecidas por el cliente y el equipo.

Anexo 3: Lecciones aprendidas de la gestión de riesgos explícita en los procesos ágiles [70].

Lecciones aprendidas	Descripción
Rol del gestor de riesgos	Es la persona responsable de asegurar que toda la documentación necesaria está al día, de que las estrategias de mitigación son revisadas al final de cada sprint, y que se identifican nuevos riesgos. Esta función se puede rotar entre los miembros del equipo y entre iteraciones.
Wiki	Un wiki funciona bien para documentar la gestión de riesgos. Todos los miembros de un equipo pueden acceder y modificar los documentos, y la mayoría de las wikis también proporcionan control de versiones para que los cambios en los documentos puedan ser rastreados fácilmente.
Equipos Pequeños de Evaluación de Riesgos de Software	Software Risk Evaluation (SRE) de SEI, es una herramienta útil para la identificación de riesgos, pero en su forma original requiere la utilización intensiva de tiempo y recursos costosos. El uso de equipos pequeños de evaluación de riesgos es una buena manera de conseguir la gestión de riesgos y establecer una lista inicial de riesgos para el proyecto. Después de la cual, toda identificación de riesgos debe hacerse de forma menos formal, en reuniones de revisiones de iteraciones con una sesión corta de lluvia de ideas.
Tareas de mitigación en Sprint Backlog	Las estrategias de mitigación deben añadirse como parte de las tareas, para integrar las actividades de mitigación de riesgos en un proyecto ágil, de lo contrario dichas tareas no serían priorizadas, podrían considerarse una sobrecarga o incluso ser ignoradas.
Estrategias de mitigación de múltiples tareas	Las estrategias de mitigación se pueden dividir en un conjunto de tareas que se incluyen a la lista de tareas que deben ser priorizadas para una iteración, permitiendo ser completadas durante el transcurso de múltiples iteraciones, además del seguimiento de mitigación de riesgos a través de mecanismos de seguimiento puestos en marcha para las tareas de iteración.
Disparador de Mitigación	Los riesgos están claramente mitigados cuando se completan las tareas de la estrategia de mitigación. El disparador de riesgos indica si los esfuerzos del equipo fueron eficaces y define las acciones que deben llevarse a cabo si en el futuro ocurre un evento específico.
Disparadores en medio de una iteración	La mayoría de los procesos ágiles incluyen una lista de tareas invariables que deben realizarse durante una iteración, para proteger al equipo de los cambios en ese período de tiempo. Si se evalúan los factores desencadenantes en el medio de una iteración, es posible que se necesite cambiar a una estrategia de mitigación de respaldo, atentando contra la agilidad y las metas establecidas en la planificación de la iteración. Por tal motivo, es adecuado definir los disparadores para que sean evaluados al final de la iteración, durante la revisión de la iteración y planear los ajustes en la planificación de la próxima iteración.

Lecciones aprendidas	Descripción
Multivotación	El equipo encontró eficaz el uso de la técnica de Multivotación, un proceso rápido que facilitó la modificación de las prioridades de los riesgos y su fácil incorporación como parte de las actividades de las reuniones regulares de planificación de la iteración. Además incluyó la opinión y participación de todos los miembros del equipo para garantizar las actividades de gestión de riesgos y la inclusión de tareas de estrategia de mitigación en equipos Sprint Backlog.
Estrategias de mitigación para nuevos riesgos	Si se identifican nuevos riesgos durante una reunión de revisión del sprint, las tareas de definir las estrategias de mitigación se incluyen en la planificación del siguiente sprint. El problema se presenta cuando se identifica un riesgo crítico, en la siguiente iteración se incluye la tarea de identificar medidas de mitigación apropiadas para este nuevo riesgo, pero no tareas para la implementación de una estrategia. La tarea de mitigación del riesgo no se producirá hasta el siguiente sprint. Es conveniente que una parte estimada del esfuerzo de planificación para el siguiente sprint sea utilizada para la mitigación real del nuevo riesgo crítico, además de la determinación de las estrategias de mitigación apropiadas para ese riesgo.

Anexo 4: Modelo propuesto para la gestión de riesgos en proyectos ágiles [71].

	Descripción	
Datos de entrada	**Datos del proyecto**	Incluye toda la información sobre el proyecto: alcance, plazos, presupuestos, requerimientos, entre otros.
	Lista de riesgos de base	Se crea tomando como base el conocimiento de otros proyectos y se actualiza con el conocimiento del proyecto actual. Inicialmente es una lista de los riesgos tradicionales.
Actividades	**Definición de la lista de riesgos**	Se define una lista de riesgos inicial, basado en la lista y los datos relativos a los riesgos del proyecto.
	Planeamiento de la gestión de riesgos	Se establece una estrategia de gestión de riesgos para todo el proyecto.
	Plan de riesgos	El plan consiste en una lista de riesgos y estrategias de minimización para cada uno de ellos.
	Monitoreo y control	Seguimiento y control que se hace de los riesgos y se aplican estrategias de minimización. En caso de nuevos riesgos, se actualiza el plan de riesgos.

Anexo 5: Modelo para la integración de la gestión de riesgos en procesos ágiles [72].

Puntos de integración	Descripción
Niveles de organización y fases de proceso	El proceso de desarrollo ágil se compone de varias fases y se extiende en varios niveles de la organización, el punto de integración consiste en identificar cuando y donde se realiza la gestión de riesgos en dicho proceso de desarrollo.
Funciones y responsabilidades	En el contexto de integración de procesos, las funciones y responsabilidades comprenden un punto importante para el intercambio de información entre los dos procesos.
Canales de comunicación	Desarrollo ágil y de gestión de riesgos son procesos intensivos de comunicación, es necesario designar canales adecuados especificando el flujo de la comunicación a nivel de toda la organización.
Aspectos del Proceso	Los procesos son variables y dinámicos, en el modelo de integración cada uno de los aspectos fundamentales de la gestión de riesgos determina la magnitud de la gestión requerida dentro del proceso.

Anexo 6: Modelo integrado para comparar la gestión de riesgos en el proceso ágil [72].

Puntos de integración	Descripción
Niveles de organización y fases de proceso	El modelo integrado cubre todo el proceso de desarrollo ágil, esto implica la integración de la gestión de riesgos en dos niveles de la organización: Negocios e Ingeniería. El nivel de negocios consiste en la fase de planificación de la visión del producto y el nivel Ingeniería consiste en la planificación de lanzamiento y hoja de ruta del producto e implementación. **Planificación de la visión del producto:** identificación y análisis de los riesgos relacionados con el negocio, tales como los riesgos de presupuesto y recursos. **Planificación de lanzamiento y hoja de ruta del producto:** comprende la identificación de riesgos, el análisis y la planificación de acciones, se trata de los riesgos de negocios y técnicos. **Implementación:** abarca principalmente la vigilancia y control de los riesgos que han sido transferidos de las fases de desarrollo anteriores a esta fase, y nuevos riesgos que se identifican de forma continua. Los riesgos son principalmente de carácter técnico.
Funciones y responsabilidades	Se identifican los roles que tienen diversas responsabilidades en materia de gestión de riesgos y su comunicación: **Foro de Gestión de Riesgos (RMF):** supervisa y coordina todos los riesgos (a nivel de negocios e ingeniería) en toda la organización y toma decisiones sobre ellos. **Business Manager:** gestiona los riesgos a nivel de negocios y puede delegar su gestión a los roles de planificación de lanzamiento y hoja de ruta del producto, manteniendo su responsabilidad hasta que los riesgos delegados sean mitigados. **Product Manager:** responsable de gestionar los riesgos en la fase de planificación del producto y planificación del lanzamiento, pudiendo delegar su gestión a otras funciones de la organización manteniendo la titularidad de los riesgos hasta que sean mitigados. **Líder y miembros del equipo:** gestionan los riesgos dentro de la fase de implementación, los miembros del equipo son responsables de los riesgos que afectan a las tareas del desarrollo que se les asigna y el líder del equipo supervisa la gestión del riesgo. Pueden decidir delegar los riesgos a otras

Puntos de integración	Descripción
	funciones de la organización.
Canales de comunicación	En el modelo integrado la gestión de riesgos se realiza durante todo el proceso de desarrollo y su gestión eficaz se basa en cómo se comunican los riesgos dentro de toda la organización. El modelo integrado identifica seis principales canales de dos vías de comunicación: (1) Planificación de la Visión del Producto - RMF, (2) Planificación de la Visión del Producto - Planificación de lanzamiento y hoja de ruta del producto, (3) Planificación de lanzamiento y hoja de ruta del producto - RMF, (4) Planificación de lanzamiento y hoja de ruta del producto - Implementación, (5) Planificación de la Visión del Producto - Implementación, y (6) Implementación - RMF.
Aspectos del Proceso	En la organización bajo estudio, se identificó el impacto de los aspectos del proceso en el modelo integrado, que se materializó en una guía de diseño de procesos: • Definir los riesgos para identificarlos y comunicarlos de manera efectiva. • Del resultado de evaluar los riesgos se identifican las acciones pertinentes para llevar adelante el desarrollo impulsado por riesgo. • El ciclo de vida del producto software ayuda en la designación de acciones apropiadas de gestión de riesgos. • La designación de roles y responsabilidades está determinada por factores como el perfil de riesgo, tipo y tamaño del proyecto. • El uso de herramientas de apoyo está determinado por factores como el perfil de riesgo, tipo y tamaño del proyecto, y distribución del equipo. • La formalidad de la información del riesgo varía en función del perfil de riesgo, tipo y tamaño del proyecto, y distribución del equipo. • La expectativa de vida del producto y el valor del negocio ayuda a

Puntos de integración	Descripción
	determinar la cantidad de procesos de gestión de riesgos necesarios. • La formalidad del proceso de gestión de riesgos está determinada por el ambiente y el contexto físico del proyecto. • La madurez y formación de la organización contribuyen en la adopción con éxito de un plan de gestión de riesgos. • Es necesario integrar el desarrollo de software con otros procesos de la organización.