TEC

**ITESM**

Administración del riesgo en desarrollos de software implementados bajo el paradigma de desarrollo ágil

TESIS

QUEPARAOBTENERELTÍTULODE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

Juan Carlos Rivera Blanco

MÉXICO,D.F.2011

**Agradecimientos**

…

**Índice**

[Capítulo 1: Introducción 4](#_Toc269321513)

[1.1 Prefacio 4](#_Toc269321514)

[1.2 Objetivo 5](#_Toc269321515)

[1.3 Justificación 5](#_Toc269321516)

[1.4 Alcance del Trabajo 5](#_Toc269321517)

[Capítulo 2: Qué es el riesgo y cómo se administra 6](#_Toc269321518)

[Capítulo 3: Situación actual de la administración de riesgos en el ámbito empresarial 6](#_Toc269321519)

[Capítulo 4: Situación actual de las prácticas de administración de riesgos en proyectos de ingeniería de software 6](#_Toc269321520)

[Capítulo 5: Problemática actual en metodologías en cascada 6](#_Toc269321521)

[Capítulo 6: Problemática actual en paradigmas ágiles 6](#_Toc269321522)

[Capítulo 7: Problemática actual en otras disciplinas 6](#_Toc269321523)

[Capítulo 8: Técnica propuesta 6](#_Toc269321524)

[Capítulo 9: Evaluación y resultados en un área de desarrollo del sector financiero 6](#_Toc269321525)

[Capítulo 10: Conclusiones y Líneas futuras 6](#_Toc269321526)

[Bibliografía 7](#_Toc269321527)

# Capítulo1: Introducción

## 1.1 Prefacio

Concluir un proyecto de desarrollo de software en tiempo y forma implica esfuerzo, recursos por parte de los participantes interesados, así como recursos materiales y tiempo. Darle término a un proyecto en tiempo significa concluir dentro de las fechas programadas, en el lapso de tiempo establecido para ese efecto, en el marco de los días designados para el proyecto. Cuando esto no sucede, se pierde dinero, competitividad, mercado, satisfacción del cliente, aceptación del consumidor, etc., y en general una disminución o insatisfacción del objetivo buscado. Terminar en tiempo es una parte fundamental y necesaria en un proyecto. Concluir en el tiempo pretendido no es todo lo que se busca, también se requiere que sea concluido en forma. La forma tiene que ver con la calidad. Entregar un proyecto en forma es entregarlo satisfaciendo los requerimientos originales, cumplir con lo solicitado, lograr lo acordado inicialmente, haber construido las peticiones del líder del producto. Hablar de la forma es una manera análoga de otras disciplinas para ejemplificar lo que se busca de un sistema. Las formas geométricas, las formas de los sólidos, la categoría taxonómica de las plantas tienen de alguna manera una representación física. Otras disciplinas más artísticas tienen que representar la forma de alguna manera física y tangible como sería el caso de la música que se percibe por los sentidos pero su forma se representa en un pentagrama. El software al ser intangible un una mezcla de ciencia y arte como DICE…, debe mostrar su forma en los resultados que genera: reportes, datos, pantallas, bytes, etc., que al ser evaluados por un cliente (otro sistema, una persona o un grupo de personas) se puede determinar si la forma es la correcta.

Para concluir en tiempo y forma un proyecto se debe seguir una serie de metodologías y paradigmas que favorezcan ese fin. Dentro de cada procedimiento o proceso de construcción del software, de manera implícita o explícita hay un manejo de los riesgos. El riesgo es un problema potencial que obstruye el objetivo último de llevar un proyecto a fase final en tiempo y forma. Un riesgo es una vulnerabilidad potencial en alguna parte del proyecto: en su contenido, en sus recursos, en el tiempo, en la administración, etc. Minimizar o eliminar esa posibilidad de que algo ocurra durante el desarrollo del proyecto, que algo o alguien obstaculice el logro del los objetivos es la tarea que ocupa a muchos profesionales de diferentes ramas no siendo la excepción la ingeniería de software.

## 1.2 Objetivo

El objetivo de este proyecto de tesis es la de analizar y proponer métodos y técnicas que favorezcan la conclusión exitosa de los proyectos de desarrollo de software analizando las situación desde la óptica del manejo y administración del riesgo, entendiendo esto último como la manera de controlar y visualizar los problemas potenciales que un proyecto de implementación de sistema conlleva.

El objetivo está acotado a la revisión del manejo de riesgos desde la perspectiva de las metodologías ágiles de desarrollo de software como son scrum, XP, cristal clear, etc. Las metodologías ágiles surgieron ante la necesidad de mejorar el proceso de desarrollo de software y de cierta manera mitigar los problemas y riesgos que más comunes aparecen en un proyecto: no concluir en tiempo, construir algo distinto a lo solicitado y que el sistema liberado tenga errores que afecten su operación.

Aunque el propósito de una evolución tecnológica fue mejorar el proceso, el cambio si redujo o eliminó varios problemas, pero provocó la aparición de nuevas variables a considerar dentro de la administración del proyecto.

En la práctica de desarrollo de software administrada por un paradigma de agilidad en general se tiende a minimizar el tema de administración de riesgos. Esto se puede notar revisando las prácticas ágiles y validando que en su mayoría la administración del riesgo está pasada por alto o llevada a cabo de manera implícita dentro del proyecto, pero sin atención especial del líder ágil o del equipo de trabajo.

Este trabajo busca indicar qué prácticas dentro de la metodología ágil en particular scrum están revelando los riesgos y proponer mecanismos que apoyen la administración y manejo de los riesgos, sin contraponer el espíritu de la práctica ágil.

## 1.3 Justificación

 La evolución de las metodologías y paradigmas de desarrollo de sistemas han obedecido a las necesidades del mercado. Las demandas generales y específicas de entregas oportunas en calidad abarcando tiempo y forma han llevado a los practicantes y al mundo académico a idear nuevas formas de trabajo que sustenten las exigencias de la industria.

 La globalización, la competencia empresarial han impulsado a las áreas de sistemas de desarrollo de software a buscar nuevas formas de ser más eficaces y eficientes en su labor.

 El fin último de las metodologías y prácticas de desarrollo de software sigue siendo concluir en tiempo y forma, y colaborar de esta manera a la competitividad de los clientes del software.

## 1.4 AlcancedelTrabajo

 Este trabajo está enfocado en analizar el manejo de los riesgos comunes en el paradigma de desarrollos ágiles que contienen la idea implícita de desarrollo iterativo e incremental. En todos los paradigmas de desarrollo existen riesgos implícitos a la metodología en sí misma y es en parte el motivo o impulso del cambio, de tal manera que en la evolución o cambio y ajustes de las metodologías los riesgos han venido cambiando. Algunos riesgos han venido a menos y otros nuevos han aparecido.

# Capítulo 2:

# Qué es scrum

Qué es scrum?

Scrum es un paradigma utilizado para desarrollo de software de tipo iterativo e incremental. Scrum está ubicada como metodología dentro de la práctica de desarrollo de sistemas bajo el paradigma de desarrollo ágil. Scrum es una de las metodologías de mayor aceptación y uso incluyendo empresas importantes a nivel mundial como Microsoft, Sun, Siemens, State Farm , IBM, Federal Reserv Bank, Yahoo (Mashking, 2005). Scrum es una de las metodologías ágiles que están mejor estructuradas y a la vez no rompen con los principios del manifiesto ágil publicados en 2001 (Sitio oficial del manifiesto ágil, 2001), sino al contrario toma los principios como suyos. En scrum se valora:

* A los individuos y las interacciones entre ellos sobre los procesos y las herramientas
* Funcionalidad completada sobre documentación comprensiva
* Colaboración con el cliente sobre negociación de contratos
* Responder al cambio sobre seguir un plan

En scrum cada iteración se denomina *sprint.* Scrum se define de manera general en los siguientes pasos (Cockburn, A., 2006):

1. Crear una lista priorizada de requerimientos llamada *backlog*

La lista priorizada de requerimientos empieza por ser una lista de requerimientos indicada directamente por el usuario o solicitante que en scrum se conocer como dueño del producto. El dueño del producto expone sus requerimientos al equipo de trabajo y los mismo se registran como cuentos del usuario. Los cuentos del usuario son referencias, indicaciones, notas para ser usadas más adelante durante el desarrollo del sistema. La diferencia más importante entre un cuento de usuario y un requerimiento es que los cuentos de usuario no son documentos finales e inamovibles, sino que son referencias a ideas que pueden ser negociables, que de hecho se desarrollarán a detalle cuando el equipo de trabajo desglose las tareas requeridas para implementar el cuento (Cohn, M., 2004). Esta técnica es más aceptada por los solicitantes pues es mucho más flexible y menos estricta, por lo que se sienten en mayor libertad para expresas sus ideas y para comunicar sus peticiones. El registro de los cuentos de usuario se realiza en una sesión de trabajo en dónde participa el equipo completo que estará trabajando durante el desarrollo del sistema y el dueño del producto. Se pueden utilizar varias técnicas para su recopilación. Una de las técnicas más usadas cualquiera que sea el formato es hacer las preguntas: ¿quién?, ¿qué?, y ¿para qué? La pregunta ¿quién?, responde a la interrogante sobre el rol, persona o grupo de personas usuarios de la característica solicitada. Es decir, quién usará el módulo, componente, funcionalidad que se solicitará. La pregunta ¿qué? es la parte de la característica en sí. Es lo similar al requerimiento, lo que se espera que el sistema realice. La última pregunta ¿para qué? sirve para dar un mayor detalle a la solicitud y establecer las bases de lo que se espera que produzca el qué. Un cuento de usuario debe contener una descripción de qué es lo que se espera que el sistema (o módulo del sistema) genere como salida. Esto será la base para establecer los criterios de aceptación del cuento y esclarece para todos los participantes la meta al construir ese cuento. Un cuento de usuario debe cumplir con ciertas características para cumplir con su función. Un cuento, en la medida de lo posible, debe ser independiente de los demás cuentos. Esto facilitará el establecer prioridades y evitará problemas de planeación. Los cuentos deben representar algo de valor para el dueño del producto. Es por ese motivo que el responsable de escribir (en el sentido de dictar) los cuentos es el mismo dueño del producto. Los cuentos deben ser cuentos y no historias épicas, es decir, los cuentos deben ser cortos y no demasiado grandes (Ambler, S. W., 2003). Una medida para conocer si el tamaño es demasiado grande es que el equipo de trabajo, los desarrolladores, son incapaces de estimar el esfuerzo requerido para su implementación. Si esto sucede ese cuento o historia épica debe desglosarse en cuentos más pequeños. Finalmente un cuento debe ser algo que se puede probar una vez completado su trabajo de construcción. Si al definir un cuento, no se pueden establecer los criterios de aceptación o los parámetros con se probará su funcionalidad o éxito, entonces el cuento es demasiado vago, abstracto o todavía no está lo suficientemente claro en la mente del dueño del producto y por consecuencia del equipo de trabajo.

Cuando los cuentos se usuario han sido expuestos y documentados, el dueño del producto establece las prioridades indicando que cuentos a su criterio ofrecen más valor para él o el negocio que éste representa (Cohn, M., 2004).

1. Planear el trabajo tomando un pequeño conjunto de requerimientos, pero los de más alta prioridad

Con esta lista priorizada de cuentos de usuario, el equipo de trabajo está listo para estimar identificar las tareas requeridas para implementar cada cuento y para estimar el esfuerzo requerido para implementar cada uno de las tareas (Pilone, D. & Miles, R., 2007). Se reúne el equipo de trabajo y usando una técnica de estimación y alguna métrica acordada estiman esfuerzos para cada cuento. Una de las técnicas de estimación mejor adoptada por la comunidad de scrum y en general por la comunidad que sigue sus prácticas con paradigmas de desarrollo ágil es la estimación de póquer. En la estimación de póquer cada miembro del equipo tiene una baraja de cartas que tiene un número que puede seguir una serie de Fibonacci o una serie exponencial. Durante la sesión de estimación el equipo escoge un cuento de usuario, lo analiza y cuando todas las dudas sobre el cuento y las tareas relacionadas están aclaradas y el equipo está listo para estimar. Al mismo tiempo cada uno de los miembros del equipo muestra una tarjeta con un número que significa la complejidad de la tarea (Agile consultants and trainers at Mountain Goat Software). Para obtener la complejidad final que tendrá la implementación de esa tarea se obtiene el promedio de las votaciones de todos los miembros del equipo y ese es el valor que tendrá la tarea. Si hay votaciones con números muy disparados, se discuten las razones de esos resultados, se despejan dudas, se revisan premisas, etc., y se realiza un nuevo ejercicio de votación. Esto se repite hasta que exista consenso y se concluye cuando todos los cuentos y las tareas han sido ponderadas.

1. Trabajar las tareas en un lapso de tiempo generalmente entre dos y cuatro semanas

El equipo de trabajo trabaja las tareas definidas para la iteración. Diariamente se reúnen en una junta para comentar sus avances. La junta es de duración breve (10 a 15 minutos) y dentro del marco del paradigma ágil se conocer como la junta de pie. Esto es debido a la naturaleza de la junta, que busca como meta no tomar demasiado tiempo y que todos participen de manera activa (Cohn, M., 2004). No es una junta dónde el líder pregunta a cada integrante del equipo sus avances, sino es una junta de participación activa dónde cada integrante responde de manera natural básicamente tres preguntas: ¿Qué hice ayer? ¿Qué voy a hacer hoy? ¿Qué impedimentos tengo para realizar mi trabajo?

Las dos primeras preguntas sobre qué hice ayer y qué voy a hacer hoy revisan la línea del tiempo de las tareas del proyecto. Por un lado, se reportan avances sobre las tareas que se están trabajando y por otro lado, se hace una revisión futura sobre lo que vendrá. La filosofía de scrum es que las tareas deben ser muy cortas, de pocas horas, de tal manera que en la junta diaria se logre obtener un avance real de la situación. El equipo de trabajo cuando reporta qué hice ayer, indica en qué tarea está trabajando (o terminó) y cuántas horas le restan para completar esa tarea. No se dice cuánto trabajo se realizó o cuántas horas se emplearon, sino cuánto falta, cuántas horas faltan para completar el trabajo.

Para responder la pregunta sobre lo que se va a realizar hoy, cada integrante del equipo toma nuevas tareas de acuerdo a sus propios criterios, capacidades y preferencias y las trabaja. Sobre esas tareas responderá la pregunta ¿qué hice ayer? al siguiente día en la junta de pie.

La última pregunta, pero no menos importante, es sobre los impedimentos que existen para poder completar las tareas que ya se están trabajando. Cada integrante del equipo expresa los problemas o impedimentos (si es que existen) que tiene por los cuales no ha podido concluir con las tareas que está trabajando. En la junta no se resuelven los problemas, sólo se expresan, se dan a conocer. El *scrum master* toma nota y busca cómo solucionar el impedimento para que ese miembro del equipo pueda continuar. La idea de fondo es que el equipo de trabajo esté totalmente concentrado en sus tareas, sin distracciones, ni presiones sobre cuestiones ajenas a la generación de código que funcione. Bajo este paradigma el *scrum master* filtra hacia el equipo cualquier distractor y entre esos se incluyen los impedimentos que puedan surgir (Highsmith, J., 2009).

1. Al finalizar la iteración, se realiza una sesión de revisión del trabajo completado y de la forma en que se realizó. El nombre en scrum es sesión de retrospectiva. El objetivo de una sesión de retrospectiva es mejorar el proceso de trabajo para tener mejores resultados en la siguiente iteración. En la sesión de retrospectiva, el *scrum master* guía al equipo a reflexionar sobre lo que ha ocurrido en las últimas semanas, repasando los hechos y resultados. La meta es recordar al equipo sobre los hechos crudos que sucedieron los últimos días (durante el *sprint*). De esta manera, sin culpar a nadie, sin acusaciones, ni personalizar los problemas, se exponen hechos en la sesión de retrospectiva, cosas que sucedieron, problemas que se encontraron, obstáculos que se presentaron o desarrollaron durante la iteración. La idea de exponerlos es que el equipo mismo de trabajo proponga las alternativas o soluciones que seguirán si algo igual o similar se vuelve a presentar en las próximas iteraciones o en iteraciones futuras. Se genera en esta reunión una base de conocimiento valioso que deberá ayudar al equipo a realizar mejor su trabajo en el futuro (Kerth, N. L., 2001).

En un equipo de trabajo que sigue una metodología de scrum existen varios roles que están clasificados en dos grandes rubros: los cerdos y las gallinas. Esta clasificación viene de la idea en que en un guisado de huevos con jamón, los cerdos están comprometidos mientras las gallinas están solamente involucradas. En otras palabras los cerdos comprometen la vida misma en el platillo y las gallinas si participan, pero su vida no está de por medio. En esta clasificación, son cerdos todos los que están comprometidos con el proyecto y son gallinas todos los que están de alguna manera involucrados, pero el proyecto no depende de su participación (Larman, C. & Vodde, B., 2008).

Cerdos:

* Dueño del producto

Es el responsable de escribir los cuentos del usuario y establecer las prioridades.

* *Scrum master*

El *scrum master* hace las veces de líder del equipo, pero en el sentido de facilitar el trabajo del equipo. El *scrum master* no asigna tareas al equipo pues el equipo se autoorganiza. Su función más importante es la de servir como barrera o filtro para aislar al equipo de trabajo de influencias externas, problemas, impedimentos, etc., y permitir que se concentren en generar código (Schwaber, K. & Beedle, M., 2002).

* Equipo de trabajo

El equipo de trabajo serán los que van a realizar el trabajo. Son las personas con las capacidades para trabajar en el proyecto. Es un equipo multidisciplinario autoorganizado.

Gallinas:

* Usuarios

Son la razón de ser del software. La gente que lo usará cuando el software sea liberado a producción. Sin usuarios el sistema y todo el desarrollo pierde sentido y razón de ser pues no habría nadie que lo use.

* Otras personas

Cualquier otro grupo de personas que pueden afectarse a favor o en contra con las actividades o resultados del proyecto.

En Scrum se utiliza dos artefactos que son básicos para el seguimiento y control de proyecto: el pizarrón de tareas (*taskboard*) y el gráfico de quemado (*burn down*).

El pizarrón de tareas es un pizarrón físico o digital que contiene las tareas del *sprint* o iteración. El pizarrón puede adoptar diversas formas, pero lo importante es que muestre el estado de las tareas, es decir, el avance sobre dónde se encuentra cada tarea: sin trabajarse, en desarrollo o terminada. Dado que scrum favorece más el software funcional antes que la documentación, el pizarrón es una herramienta simple (no compleja) que de manera muy visual muestra a todos (cerdos y gallinas) el estatus del proyecto (Highsmith, J., 2009).



*Figura* 1. Pizarrón de Tareas de Scrum.

En la Figura 1, se muestra un ejemplo de un pizarrón de tareas de scrum con cuatro columnas. La primera columna de la izquierda contiene los cuentos de usuario que se estarán trabajando durante la iteración; en la segunda columna van todas las tareas que se realizarán para implementar los cuentos de usuario; en la tercera columna se ponen las tareas que el equipo está trabajando, es decir, las tareas que están en progreso, los ítems que se están construyendo; y por último en la columna de la derecha van las tareas completadas, las tareas que el equipo concluyó, que fueron codificadas y probadas. En los renglones de la cuadrícula del pizarrón van los cuentos de usuario y sus respectivas tareas a realizar. Lo que no aparece en este listado, no se trabajará en el *sprint.*

Durante la junta diaria, cada miembro del equipo de trabajo toma una tarea del pizarrón de la columna “Por hacer” y la mueve a la columna “En progreso” para indicar a los demás integrantes del equipo que ha decido trabajar esa tarea y le estará dedicando esfuerzo y tiempo. En la próxima junta de pie o junta diaria, cada miembro del equipo, si terminó, mueve su tarea a la columna “Terminadas”. Terminar una tarea significa que el desarrollo ha concluido y que todas las pruebas se han ejecutado de tal manera que la tarea de ninguna manera puede volver a “En progreso” o “Por hacer”. De esta manera, diariamente, durante las juntas de pie el equipo va avanzando las tareas de estado. La metodología es flexible y un miembro del equipo no debe necesariamente esperar a la junta de pie para tomar nuevas tareas o mover las que han tomado de columna de acuerdo a su avance, sino que puede realizarlo a discreción durante la jornada laboral, siempre y cuando indique esos progresos en la siguiente junta diaria.

Si el equipo de trabajo por cuestiones de espacio o cualquier otra razón de importancia para el equipo mismo, decide tener el pizarrón de tareas en formato digital con algún archivo de MS Excel o cualquier otra herramienta, lo importante del concepto es tener la actualización diaria del documento electrónico por cada miembro del equipo y por ende que ese documento o archivo esté siempre editable para todos los miembros del equipo con rol de “cerdo” y visible para todos cerdos y gallinas.

El gráfico de quemado es la representación gráfica de las horas faltantes para completar las tareas del *sprint*. En principio en scrum no se mide el trabajo realizado, las horas empleadas ni las horas gastadas en las tareas, sino las horas que hacen falta por cada tarea para completarse. Las horas que el equipo de trabajo estimó en un inicio para completar una tarea, son las horas que faltan de trabajo para completarla. Si el trabajo empezó a realizarse, en la junta diaria cada miembro del equipo irá reportando cuántas horas faltan para completar las tareas que cada uno tomó hasta llegar a cero horas y mover la tarea de columna en el pizarrón de tareas. El progreso de las horas faltantes de trabajo reportadas en la junta diaria se muestra en el gráfico de quemado (Cohn, M., 2004).



*Figura* 2.1 Gráfico de quemado indicando trabajo remanente. Vista en el día uno del *sprint*.

Como se muestra en la figura 2, en el eje de las X o eje horizontal en el gráfico de quemado se muestran los días que durará el *sprint*. En el eje de las Y o eje vertical del gráfico se muestra las horas totales resultado de la suma de las horas estimadas de trabajo a realizar por cada una de tareas correspondientes a los cuentos de usuario que se estarán trabajando durante la iteración.

Dado que el gráfico representa las horas que faltan de trabajo para completar todas las tareas, el día uno de trabajo habrá un indicador en la columna del “día 1” en la parte superior del eje de las Y, dónde está el total de horas faltantes, pues ese día se está empezando la iteración y hace falta emplear todas las horas del *sprint* para concluirlo. El “día 2” el indicador irá en el punto que le corresponda con relación a las horas que falten dado el avance que el equipo de trabajo tuvo en el primer día de trabajo. En los siguientes días se irá reflejando las horas que faltan por trabajar de la suma de todas las tareas del equipo con la intención que el último día del sprint queden cero horas por trabajarse.



*Figura* 2.2 Gráfico de quemado con línea base o tendencia esperada.

La figura 2.2 muestra una línea ideal que es similar a la línea base de otras metodologías. La línea que está señalada con un etiqueta como “Tendencia esperada” con una pendiente negativa que va del 72 al cero en el eje de “Horas totales de la iteración” muestra la línea ideal que el proyecto debe seguir en esta iteración. Esta línea indica cómo deberían consumirse las horas durante el *sprint*. Esta es la línea que el equipo de trabajo tiene como meta de su trabajo. Cuando el equipo se reúne en su junta diaria y actualiza las horas restantes de sus tareas, la línea que aparece horizontal como “Trabajo remanente” idealmente irá acercándose a la línea de la “Tendencia esperada”. Cuando un proyecto va atrasado en tiempo, la línea de “Trabajo remanente” estará por encima de la línea de “Tendencia esperada”. Cuando un proyecto va adelantado en el tiempo, la línea de “Trabajo remanente” estará por debajo de la línea de “Tendencia esperada”. Lo que sucede con frecuencia en un *sprint* es que la línea de “Trabajo remante” hace un efecto de curvas siguiente la línea de “Tendencia esperada” indicando que algunos días el equipo de trabajo produjo más producto y en otros días, menos afortunados, se produjeron menos resultados favorables.



*Figura* 2.3 Gráfico de quemado con línea indicador de tendencia.

El gráfico de quemado se puede enriquecer con una línea de tendencia. La línea de tendencia indicada con una etiqueta en el gráfico como “Tendencia actual” indica la velocidad con la que el equipo de trabajo está completando las tareas y si las cosas siguen de ese modo o a ese ritmo en qué tiempo estarían terminando todas las tareas de la iteración. Con la línea de tendencia todo el equipo está al pendiente que los resultados de su esfuerzo y si hay que realizar algún ajuste a la manera de trabajar o a la forma de trabajo que favorezca el logro del objetivo común: concluir en tiempo y forma.

La metodología de Scrum cuenta con una certificación avalada por el *Scrum Alliance*. La certificación es como *scrum master*. No todos los practicantes de la metodología están de acuerdo con que exista una certificación e inclusive es criticado por equipos y personas que siguen metodologías ágiles para el desarrollo de software. La crítica es que una certificación encasilla la práctica en un marco rígido y quita flexibilidad. A entender de esta postura, las metodologías ágiles deben tener mayor libertad de acción y libertad para adoptar o dejar artefactos, prácticas, etc., y de esta manera utilizar lo más conveniente para cada situación, proyectos y equipo de trabajo. Sin embargo, la certificación otorgada por el Scrum Alliance ofrece cierta certidumbre que la persona certificada entiende y está capacitada para seguir la metodología y guiar a un equipo de trabajo en la práctica de scrum.

# Qué es el riesgo y cómo se administra

# *2.1 Definición*

# De acuerdo a la definición el diccionario de computación de la universidad de Oxford, un riesgo es la cantidad resultado de la probabilidad que un problema particular ocurra y la magnitud de la consecuencia de los efectos indeseables del problema. De manera informal, el término riesgo se usa para indicar la probabilidad que un problema ocurra (Daintith, E. J. &Wright, E., 2008).

# Jalote (2000) define el mismo concepto como exposición al riesgo y plantea una fórmula para explicarlo. Si Probabilidad(R) es la probabilidad de que un riesgo R ocurra y Perdida(R) es el total de perdida a la que se incurre si el riesgo se materializa, entonces la exposición al riesgo, RE, para el riesgo dado se define con la siguiente ecuación:

# RE(R) = Probabilidad(R) x Perdida(R)

# Dentro del ámbito de administración de proyectos, el riesgo es la probabilidad de que algo ocurra de manera desfavorable para el proyecto. El riesgo es la potencialidad que una vulnerabilidad ocurra y se genere un perjuicio para el proyecto. Un riesgo es un problema que no se ha dado o que no se ha presentado en un proyecto, pero está latente y si no se toma algún tipo de acción puede presentarse y afectar el presente y futuro de un proyecto.

# En todas las disciplinas y áreas de trabajo se requiere administrar o manejar los riesgos de los proyectos o tareas a desarrollar para poder tener éxito en el logro de los objetivos y metas. Si los problemas potenciales o riesgos de un proyecto no se toman en cuenta, no se les da atención, no se consideran o no se atienden, el proyecto tiene tanta posibilidad de fracasar como los riesgos de ocurrir.

# El manejo de los riesgos no es una ciencia exacta ni existe una solución o receta general para su tratamiento. Es una disciplina que está fuera de las ciencias exactas, sus resultados no pueden ser medidos o documentados mediante el método científico dónde la repetición es invariable dadas las mismas condiciones de entrada. En el caso de los riesgos y su administración, no hay fórmulas universales que nos den soluciones únicas y definitivas, es por ello que expertos de la materia y de diferentes disciplinas han hecho sus aportaciones en base a casos de estudio, experiencias y conocimiento académico para procurar orientar a los líderes y administradores de proyectos a solventar y adoptar prácticas que lleven a un mejor manejo de los riesgos.

# Los riesgos pueden ser clasificados de diversas maneras: tipos, elementos, severidad, ocurrencia, impacto, etc. La clasificación se hace de acuerdo a lo que se pretende medir o la meta que se busca cubrir. Cuando una organización está recabando, documentado y buscando acciones para mitigar riesgos puede clasificar por ingresos, imagen, operación, etc., que van de acuerdo a las políticas y metas que cada compañía tiene. Cada negocio, organización, grupo, consorcio o compañía determinará los aspectos que cuidará mayormente de acuerdo a su giro de negocio. Por ejemplo, una empresa del sector financiero que da servicio a bancos y maneja información sensible de carteras de crédito buscará encontrar y minimizar los riesgos que se refieran a la seguridad de la información resguarda. Un problema de filtración, fuga o alteración de la información causaría una desconfianza en sus clientes que podría acabar con el negocio.

# En los proyectos de desarrollo de software también hay diversos factores que pueden afectar la conclusión de la implementación de un sistema computacional como pueden ser: las definiciones, recursos tecnológicos, recursos humanos, la definición de prioridades, la multiplicidad de las tareas y proyectos por mencionar algunos.

# Dependiendo de la metodología de desarrollo utilizada el riesgo se administra de diferente manera. En proyectos administrados que siguen los conceptos y recomendaciones del Instituto de Administración de Proyecto (Project Management Institute, PMI) existe un apartado completo que incluye técnicas y documentos para llevar a cabo la administración del riesgo. En un proyecto que se implementa con el Proceso Unificado Racional (Rational Unified Process, RUP) hay una reserva de documentos y formatos que el líder del proyecto debe implementar para mitigar o administrar los riesgos. En un proyecto que se practica con un paradigma de desarrollo de software ágil, el riesgo en lo general no es explícitamente administrado, aunque la metodología en sí busca minimizar riesgos que se presentan en otras metodologías.

# *Cómo afectan*

# Cada riesgo representa una amenaza para el logro exitoso de un proyecto. Si los riesgos se materializan y sus consecuencias o daños son importantes o graves, pueden mal lograr o definitivamente aniquilar proyectos completos. Esto por su puesto trae consecuencias desfavorables para las organizaciones y compañías que han invertido tiempo, recursos y esfuerzo por desarrollar proyectos o sistemas computacionales para sustentar su operaciones ya sea en sus procesos internos, la venta de productos o mantenimiento de su competitividad en el mercado o sector al que pertenezca.

# Se ha observado que cerca del 15 por ciento de todos los proyectos estudiados desde 1977 (quinientos proyectos) han fallado: algunos fueron cancelados, abortados, postpuestos o entregaron productos que nunca se usaron. En proyectos grandes las estadísticas son todavía peores, el veinticinco porciento de los proyectos realizados en los últimos veinticinco años de trabajo o más han fallado en completarse. (DeMarco, T. & Lister T., 1999)

# La medición del impacto que los riesgos pueden tener en las organizaciones es complejo de establecer. Se pueden realizar ejercicios de proyección, estimación, pronosticación, etc., aplicando diversas técnicas que al final pudieran derivar en conclusiones demasiado abstractas y sin mucho valor real para las áreas, personas o instituciones encargadas de administrar los riesgos. Por otro lado, es importante para los directivos y tomadores de decisiones conocer los números resultado de la medición del riesgo para a su vez determinar los esfuerzos que se requerirán implementar para su solución.

# Para solventar esta situación, en muchos casos cuando la medición cuantitativa no es factible o no entrega resultado reales, lo que se realizar es obtener métricas cualitativas. Una de las maneras para obtener las métricas cualitativas es tomar la historia, conocimiento, experiencia previa de la misma organización o de otras organizaciones del mismo sector con la finalidad de poder pronosticar en base a experiencia los efectos o magnitudes que la capitalización de un riesgo puede traer como consecuencia.

#

# Tabla 1. Proyectos de tecnología de información no exitosos (Charette, R. N., 2005)

# En la tabla 1 se muestra un listado de compañías que se han impactado por temas relacionados en el uso de sus sistemas computacionales. En la columna “Año” está indicado el año en que la compañía reportó el evento, en la segunda columna, “Compañía”, está indicado el nombre de la compañía que se afectó negativamente indicando el país entre paréntesis en caso de no ser Estados Unidos de Norte América; por último, en la tercera columna se muestran los resultados desfavorables en términos de dinero en dólares americanos que derivan de pérdidas para las compañías y una breve descripción de la causa por la cual se da la pérdida.

# Charette (2005) indica que una de las causas por las cuales los proyectos de software fallan es por la falta de administración del riesgo, dejando claro que no es una sola la causa por la que los proyectos fallan. La lista que Charette (2005) indica como causas por las cuales un proyecto falla es la siguiente:

# Metas irrealistas o sin articular

# Estimaciones imprecisas of recursos necesarios

# Requerimientos del sistema mal definidos

# Reportes pobres del estatus del proyecto

# Riesgos sin administrar

# Comunicación pobre entre los clientes, desarrolladores y usuarios

# Uso de tecnología inmadura

# Inhabilidad para manejar proyectos complejos

# Malas prácticas de desarrollo

# Pobre administración del proyecto

# Políticas de los involucrados en el proyecto

# Presión Comercial

# *Cómo abordan los riesgos cada metodología*

# Las metodologías y prácticas de desarrollo se han enfocado a tratar de socavar áreas de oportunidad de metodologías ya existentes o a solventar necesidades que surgen del ambiente cambiante en que desarrollan las organizaciones y compañías. A continuación se listan las metodologías, técnicas o paradigmas de desarrollo de software más representativas y cómo abordan o no el tema de riesgos y su administración.

# *Rational Unified Process (RUP)*

# Uno de los principios de RUP es el de atacar los mayores riesgos desde el inicio del proyecto y continuar durante la vida del proyecto con su seguimiento. De no hacerlo de este modo los riesgos te atacarán a ti (Gilb, T. & Finzi, S., 1988). RUP busca con el modelo iterativo, atacar riesgos desde las primeras fases del proyecto que en métodos de cascada por su naturaleza parecen después en el tiempo cuando es tarde y algunas consecuencias desfavorables para el proyecto ya se han presentado.

# El RUP enfoca sus primeras etapas, el inicio del proyecto en establecer las bases: comprender la problemática, revisión de la tecnología, delimitar el alcance y ámbito del proyecto, establecer una línea base del proyecto completo y eliminar los riesgos críticos del proyecto. Una de las salidas de las primeras fases de implementación de RUP es el artefacto llamado plan de fase. En este documento están planeadas el número de iteraciones que tendrá el proyecto, las fechas clave para cada iteración así como la documentación de riesgos.

# Una de las razones por las que RUP realiza estos pasos desde el inicio del proyecto es con la finalidad de establecer una arquitectura robusta que soporte todo el proyecto y las iteraciones con la menor cantidad posible de cambios en ésta. De esta manera, RUP hace su propuesta que estas acciones mitigan riesgos que más adelante en el proyecto pueden ser inmanejables o demasiado costosos para el éxito del mismo (Jacbson, I., Booch, G. & Rumbaugh, J., 1999).

# El modelo de cascada

El modelo de cascada plantea una serie de pasos para lograr el éxito en los proyectos de desarrollo de software o proyectos de tecnologías de información. Lo que se busca es entregar todo lo que el cliente quiere en tiempo y dentro del presupuesto asignado.

El modelo de cascada plantea como premisa dedicar todos los recursos necesarios de tiempo y esfuerzo en la obtención y clarificación de los requerimientos del sistema a construir y después diseñar la solución completa. La propuesta es a que una vez ejecutados con precisión estos dos primeros pasos no habrá mayores cambiar en el proyecto. Parte de esta filosofía es documentar perfectamente estas fases y recopilar las autorizaciones de los usuarios y/o solicitantes para evitar cambios futuros. El paradigma de cascada asume que es posible identificar completamente los requerimientos desde el inicio del proyecto y que esos requerimientos no cambiarán en el tiempo. Básicamente el desarrollo es transformar en un lenguaje de programación lo plasmado en un lenguaje humano.

El modelo de cascada propone la minimización de los riesgos de un proyecto dejando claro el alcance del proyecto desde un punto de vista del usuario y también desde un punto de vista técnico al inicio, en las primeras etapas del proyecto. Si todo está completamente definido y entendido por el equipo de trabajo y nada de lo requerido cambiará durante de la vida del proyecto ni se alterará el alcance entonces los mayores riesgos de un proyecto de tecnología de información están cubiertos (Castro, R., 2004).

# La metodología de Espiral

# La metodología de espiral creada por Boehm en 1986 es un modelo orientado al riesgo desarrollando desde las primeras fases diseños y prototipos que establezcan bases sólidas y requerimientos claros para el resto del desarrollo del software. La metodología busca aprovechar las ventajas de los conceptos de análisis de arriba-abajo y de abajo-arriba.

# La metodología dedica una fase en particular especialmente a detectar los riesgos del proyecto. La idea de un diseño preliminar que contemple todas las posibles alternativas busca minimizar los riesgos de indefiniciones futuras en los requerimientos del sistema. En el paradigma de espiral el manejo o administración de riesgos se circunscribe a clarificar y detallar requerimientos desde etapas muy tempranas del proyecto. La propuesta es contar con diseños y prototipos que establezcan las bases del resto del proyecto y así tener desde un inicio la visión de lo que se espera tanto de lado del solicitante como del equipo de trabajo.

# La metodología no propone algún mecanismo adicional en relación a otros riesgos o su manejo. Su enfoque es básicamente en el riesgo común de los proyectos de software: “requerimientos que no son claros” que si se capitalizan terminan en circunstancias adversas en cuanto a tener un producto o productos de software que realizan funciones o tareas distintas a lo considerado por el solicitante. Cuando esto sucede, el cliente no está satisfecho con la entrega e inclusive los proyectos no se concluyen pues no se logra la meta de tener sistemas que funcionen y funcionen para los fines que fueron solicitados (Bohem, B., 1986).

# PMI

# PSP (Personal Software Process)

# TSP (Team Software Process)

# Para el TSP como en otros paradigmas la administración del riesgo

# CCM CapabilityMaturityModel

El CMM establece un apartado de administración de riesgos en la fase planeación de los proyectos. Desde la primera etapa del proyecto el CMM indica que se debe establecer una administración de riesgos y dicta actividades que se deben ejecutar estableciendo el orden y taxonomía de los riesgos.

Dentro de la taxonomía propuesta por el CMM se clasifica y enfatiza sobre dos componentes clave en la administración del riesgo que son: la identificación y el control de los riesgos. La taxonomía completa se muestra en la siguiente figura 3.



Figura 3. Actividades de administración de riesgo (Jalote P., 2000)

En primer lugar el CMM indica los componentes clave de la clasificación u organización de los riesgos: evaluación y control. El primer punto de control es la identificación. La identificación es la parte de visualizar los riesgos para los cuales el plan de riesgos se enfocará. Esto es con el fin de estimar el esfuerzo y recursos que se deberán dedicar para mitigar o eliminar la situación que si se presentan traerían consecuencias desfavorables para el proyecto. La idea de identificar y estimar está desglosada en tres apartados que son: prospectar, analizar cada punto visualizado y establecer una prioridad sobre lo analizado. Las prioridades determinan los esfuerzos que se emplearán o dónde, en qué puntos se debe emplear esfuerzos extras de tal manera que se maximicen los resultados del esfuerzo empleado a la administración de riesgos. Para priorizar los riesgos se debe estimar dos cosas importantes: la probabilidad que cada riesgo ocurra y las consecuencias para el proyecto si el riesgo se capitalizara. El CMM propone utilizar la fórmula valor ganado multiplicando la probabilidad de ocurrencia con el valor de la pérdida.

Una vez concluida toda la etapa de identificación se decide el plan de cómo o qué hacer con cada riesgo. El CMM establece que sólo el grupo pequeño de riesgos que ocupen el lugar de mayor prioridad serán atendidos en el proyecto. Una posibilidad para la implementación del plan de administración de riesgos es tratar de prevenir o evitar las acciones que provocarían un riesgo. Si es el uso de cierta tecnología que el equipo de trabajo no conoce, la estrategia sería optar por otra que sea del dominio del equipo, así se evita el riesgo. Desafortunadamente no todos las acciones se pueden evitar. Podría ser que esa tecnología sea requerida para la implementación de una parte del proyecto de software que es indispensable y motivo del proyecto.

La implementación del plan de riesgos y el plan en sí deben ser monitoreados con la finalidad de estar al pendiente de las estrategias para mitigar los riesgos. Durante toda la ejecución del plan se debe actualizar el status de cada riesgo para mantener el control del riesgo y del plan.

# Ágil

# En el caso del paradigma de scrum la administración de riesgos se propone realizar como un mecanismo continuo con una comunicación abierta entre los integrantes del equipo que permita mostrar diariamente cualquier impedimento que se encuentre en el proyecto o las actividades de los miembros del equipo para ser mitigadas de inmediato. La propuesta es identificar los riesgos en las juntas diarias o *daily scrum* en el análisis de la pregunta “¿qué impedimentos tengo para realizar mis tareas?” En scrum los problemas estarán expuestos hasta que hayan sido solucionados. (Schwaber, K., 2007)

# El rol del *scrum master* es de suma importancia en el tratamiento de los riesgos ya que es el quién tendrá que dar seguimiento a todas las incidencias o riesgos detectados. Su labor será la de concentrar los impedimentos en el backlog de impedimentos y velar por su resolución.

# Bajo esta óptica la administración de los impedimentos es la administración de los riesgos. Sin embargo, la propuesta actual de scrum está únicamente atacando lo que ya está ocurriendo como impedimento y no proyectando los posibles problemas. Aunque es una manera funcional para el equipo que el *scrum master* filtre los impedimentos y se haga cargo de resolver para que el equipo continúe concentrado en generar software que funcione, hay un universo de situaciones o posibles impedimentos que no se están visualizando ni reportando por ningún miembro del equipo, pues sólo se reporta lo que ya está sucediendo. Es decir, la respuesta a la pregunta “¿qué impedimentos tengo?” es un asunto presente y no futuro. Si consideramos la definición de riesgo como algo que todavía no ocurre sino que está en posibilidades de suceder, el scrum no está administrando riesgos sino problemas.

# Considerando esto, algunos practicantes de scrum han ampliado la pregunta a “¿qué impedimentos tengo o voy a tener?” y de esta manera tener un panorama más amplio sobre las tareas siguientes (Albaladejo).

# Capítulo 3: ¿QUÉ? Modelo de extensión de scrum para el manejo de riesgos

# Una combinación del Impediment backlog con una gráfica de top of mind de burbujas visible para todo el equipo.

# Podría ser dentro del taskboard es indicar las tareas con un color de alerta (rojo). Si va en otro artefacto como el backlog de impedimentos la propuesta es agregar columnas que reflejen el porcentaje o probabilidad de ocurrencia y también el efecto adverso que causaría de presentarse el riesgo.

# Capítulo X: Conclusiones y Líneas futuras

# Bibliografía

"risk"  *A Dictionary of Computing*. Ed John Daintith and Edmund Wright. Oxford UniversityPress, 2008.*Oxford Reference Online*. Oxford UniversityPress.  ITESM.  26 January 2011  <http://0-www.oxfordreference.com.millenium.itesm.mx/views/ENTRY.html?subview=Main&entry=t11.e4523>

http://www.martinfowler.com/articles/newMethodology.html

# Jalote, Pankaj. CMM in Practice Processes for Executing Softwate Projects at Infosys.SEI Series in software engineering. 2000.

# Mashkin, Berteig. Mayo 12, 2005.Scrum Gathering May 2005 in Boston – Rough Notes (Agile Advice).http://www.agileadvice.com/archives/2005/05/scrum\_gathering.html (Accesado en Febrero 5, 2011)

# Sitio oficial del manifiesto ágil. 2001. Manifesto for Agile Software Development. http://www.agilemanifesto.org/. (Accesado el 5 de febrero de 2011).

# Schwaber, Ken (2007) The Enterprise and Scrum. Microsoft Press.

# Cockburn, Alistar. 2006. Agil Software Development: The Cooperative Game. Addison-Wesley Professional.

# Cohn, Mike. 2004. User Stories Applied: For Agile Software Development. Addison-Wesley Professional.

# Pilone, Dan & Miles, Russ. 2007. Head First Software Development. O’Reilly Media, Inc.

# Ambler, Scott W. 2003-2009.Introduction to User Stories.Ambysoft Inc. http://www.agilemodeling.com/artifacts/userStory.htm.(Accesado el 8 de febrero de 2011).

# Agile consultants and trainers at Mountain Goat Software. Planning Poker. Agile NDA. http://www.planningpoker.com/. (Accesado el 8 de febrero de 2011).

# Highsmith, Jim. 2009. Agile Project Management: Creating Innovative Products. Addison-Wesley Professional.

# Kerth, Norman L. 2001. Project Restrospectives: A Handbook for Team Reviews. Dorset House.

# Larman, Craig &Vodde Bas. 2008. Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum. Addison-Wesley Professional.

# Castro Gil, Robin Alberto. 2004. Estructurabásica del procesounificado de desarrollo de software. Universidad ICESI Sistemas&Telemática.

# Jacobson, Ivar.,Booch, Grady &Rumbaugh, James. 1999. The Unified Software Development Process. Rational Software Corporation. Addison-Wesley.

# Boehm, Barry. 1986. A Spiral Model of Software Development and Enhancement. ACM SIGSOFT Software Engineering Notes. 11(4):14-24, August 1986

# Schwaber, Ken & Beedle, Mike. (2002). Agile Software Development with Scrum. Upper Saddle River, NJ: Prentice Hall

# Gilb, Tom & Finzi, Susannah. (1988). Principles of software engineering management. Addison-Wesley Pub. Co. <http://books.google.com.mx/books?id=w55QAAAAMAAJ&q=Principles+of+Software+Engineering+Management&dq=Principles+of+Software+Engineering+Management&hl=es&ei=DURTTazJGYf2tgPtrpDhAw&sa=X&oi=book_result&ct=result&resnum=1&ved=0CCQQ6AEwAA>

# Albaladejo, Javier. Proyectos Agiles. Ingent. http://www.proyectosagiles.org/que-es-scrum. (Accesado el 9 de febrero de 2011).