UNIVERSIDAD CENTROCCIDENTAL

"LISANDRO ALVARADO"

**MODELO INTEGRADO PARA LA EVALUACIÓN DE LA MADUREZ DE LAS CAPACIDADES DE LOS MÉTODOS ÁGILES DE LOS PROYECTOS DE DESARROLLO DE SOFTWARE**

Neneskha A. Salas Albahaca

Barquisimeto, 2018

UNIVERSIDAD CENTROCCIDENTAL “LISANDRO ALVARADO”

DECANATO DE CIENCIAS Y TECNOLOGIA

COORDINACIÓN DE POSTGRADO

**MODELO INTEGRADO PARA LA EVALUACIÓN DE LA MADUREZ DE LAS CAPACIDADES DE LOS MÉTODOS ÁGILES DE LOS PROYECTOS DE DESARROLLO DE SOFTWARE**

Trabajo presentado para optar al grado de Magister Scientiarium en Ciencias de la Computación

Mención Ingeniería de Software

Por: Neneskha A. Salas Albahaca

Barquisimeto, 2018

**ÍNDICE GENERAL**

Resumen v

Introducción 1

**CAPÍTULO I, EL PROBLEMA**

Planteamiento del Problema 4

**Objetivos de la Investigación**

Objetivo General 11

Objetivo Específicos 11

Justificación y relevancia de la Investigación 13

Alcance de la Investigación 15

**CAPÍTULO II, MARCO TEÓRICO**

Antecedentes de la Investigación 17

**Bases Teóricas**

Modelos de Proceso de Desarrollo de Sistema de Software 21

Enfoque ágil de los proyectos de Desarrollo de Sistemas de Software 23

Principios del Manifiesto Ágil de Desarrollo de Software 24

Noción disciplinaria de los equipos en el Proceso de Desarrollo de Sistemas de software 25

Métodos Ágiles de Proyectos de Desarrollo de Sistemas de Software 28

Gerencia de la Calidad como marco para la medición de la Madurez de las Capacidades de los Procesos de Proyectos Desarrollo 37

Capacidad del Proceso de Desarrollo de Sistemas de Software 39

Modelos de Madurez para la evaluación de las Capacidades de los Procesos de Desarrollo de Sistemas de Software 41

Definición de Términos Básicos 42

**CAPÍTULO III, MARCO METODOLÓGICO**

Naturaleza de la Investigación 44

Diseño de la Investigación 45

Procedimiento de la Investigación 45

Referencias Bibliográficas .................................................................................51

**INDICE DE FIGURAS**

Figura 1. Método AUP .................................................................................................... 35

Figura 2. Método Scrum....................................................................................................37 Figura 3. Método XP.........................................................................................................39

Figura 4. Modelo general de un ciclo de investigación en ciencia del diseño..................51

Figura 5. Proceso genérico de la Ingeniería de Método Situacional.................................52

Figura 6. Enfoque basado en Extensión de la ingeniería de método situacional..............54

**MODELO INTEGRADO PARA LA EVALUACIÓN DE LA MADUREZ DE LAS CAPACIDADES DE LOS MÉTODOS ÁGILES DE LOS PROYECTOS DE DESARROLLO DE SOFTWARE**

**Autor:** Neneskha Salas Albahaca

**Tutor:** Jean Carlos Guzmán

**RESUMEN**

La Ingeniería de Software es una disciplina que trata los aspectos concernientes al desarrollo de sistemas de software complejos y de calidad, que requiere para su construcción un equipo de trabajo, procesos rigurosos, sistemáticos y controlables (modelos y métodos) y la utilización de herramientas. Es un campo dentro de las ciencias de la computación, basado en un conjunto de métodos, técnicas y herramientas, y en principios aceptados por la comunidad científica que tienen como propósito crear productos o servicios únicos, determinados por el alcance, el costo y el tiempo. Actualmente, la disciplina emplea un enfoque ágil basado en un modelo de proceso iterativo e incremental fundamentado en el ciclo de vida adaptativo que admite efectividad, rendimiento y productividad, desde una perspectiva organizacional-gerencial tecnológica. Sin embargo, la madurez de estos procesos de desarrollo en la industria del software, se encuentran en un estado crítico, en vista que se desconoce la madurez y las capacidades, que aportan estos métodos de desarrollo a las organizaciones que siguen este enfoque. Esta investigación propone el diseño de un modelo integrado para la evaluación de la madurez de las capacidades de los métodos ágiles de los proyectos de desarrollo de sistemas de software bajo el enfoque de investigación ciencias del diseño con el apoyo del método general para la investigación en la ciencia del diseño y el método genérico de la ingeniería de método situacional bajo el enfoque basado en extensión.

**Palabras claves**: Madurez y Capacidad, Modelo de Proceso, Métodos Ágiles, Proyectos de Desarrollo, Sistemas de Software.

**INTRODUCCIÓN**

El vertiginoso desarrollo de las Tecnologías de Información y Comunicación (TIC) ha transformado significativamente las formas tradicionales de producir e intercambiar bienes y servicios. La incorporación de estas herramientas a las áreas neurálgicas de las organizaciones se ha convertido en un punto focal para la gerencia, dado que permiten alcanzar altos índices de efectividad, rendimiento y productividad, agregando valor y nuevas capacidades desde una perspectiva organizacional-gerencial tecnológica. En este sentido, para lograr el desarrollo de un producto de software en particular, en el tiempo, el presupuesto y el cronograma, es necesario que se considere como estrategia el enfoque ágil de proyectos de desarrollo de software.

De lo anterior, se infiere que estas variables se constituyen en elementos de riesgos que deben ser necesariamente canalizados con antelación y mitigados durante la ejecución del proyecto. Para Guzmán (2010, p. 8), un alto porcentaje de proyectos de desarrollo de software ostentan fracasos financieros debido que “…no se encuentran orientados o alineados hacia los principios agiles definidos en el Manifiesto Ágil y no se considera al usuario que utilizará el producto derivado de ese proceso”. De esta aseveración, se plantea que el éxito de un proyecto de desarrollo depende de la satisfacción del cliente, quien es el sujeto o la entidad, que finalmente invierte los recursos financieros necesarios para su desarrollo y al usuarios que es el sujeto que finalmente lo utilizará; esto conlleva a exigencias relativas al cierre del proyecto en el tiempo establecido y dentro del presupuesto inicialmente previsto.

De allí que, los “…problemas reales que se suscitan en las organizaciones y que enfrenta el responsable del proyecto, cuando intenta satisfacer las necesidades y requisitos de los sujetos sociales, se enmarca en la complejidad atinente a la consecución de las actividades y tareas” (Guzmán, 2010). En este contexto, es imperioso resaltar que algunos de estos problemas han sido aceptados como inevitables, en la realidad de estas organizaciones.

Aunado a este hecho, la comunidad científica ha buscado soluciones específicas a estos contratiempos; sin embargo, existen preocupaciones con respecto a su efectividad, en vista que esta problemática se repite de manera constante, en contextos diferentes. Empero, los proyectos de desarrollo de software no se eximen de esta problemática, dado que cada vez más las comunidades organizacionales plantean estrategias para superar las limitaciones organizativas, culturales y metodológicas, que buscan solventar esta situación, en medio de exigencias que estriban en que el producto resultante ostenta ciertas características de calidad. Es necesario resaltar que la calidad del proceso de desarrollo de software incide en la calidad del producto resultante.

Lo que antecede, suplica el diseño de un modelo de madurez para la evaluación de las capacidades de los métodos de los proyectos de desarrollo de sistemas de software en las organizaciones de la Industria del Software, por ende, se requiere de un conjunto de directrices necesarias para su materialización. Un apropiado proceso de proyectos de desarrollo admite la generación de valor y de nuevas capacidades en las organizaciones; pues de ello depende que las empresas de desarrollo e innovación alcancen sus objetivos y satisfagan sus metas, haciendo uso eficaz de los recursos, con plena convicción de generar productos o servicios, sin alterar las variables: costo, tiempo y alcance.

Esta investigación hace referencia a esta problemática, la cual está afectando la efectividad, el rendimiento y la productividad de las organizaciones que integran la Industria del Software, que se presentan con anuencia de restricciones en las citadas variables, debido a la carencia carecen de un modelo que precise el nivel de madurez y las capacidades generadas a través de la implementación de métodos ágiles. Está estructurada por tres (3) capítulos que se mencionan a continuación:

Capítulo I, muestra la problemática que se presenta por la carencia de un modelo de madurez para la evaluación de las capacidades de los métodos ágiles de proyectos de desarrollo en la Industria del Software, donde se propone el diseño de un modelo de madurez con la finalidad de mejorar su rendimiento y la generación de nuevas capacidades.

Capítulo II, describe el marco teórico conformado por los antecedentes y las bases teóricas o fundamentos que sustentan la investigación, entre las cuales se encuentra: Modelos de Proceso de Desarrollo de Sistemas de Software; Principios del Manifiesto Ágil de Desarrollo de Software; Noción disciplinaria de los equipos en el proceso de desarrollo de software; Métodos Ágiles de Proyectos de Desarrollo de sistemas de Software: Gerencia de la Calidad como marco para la medición de la Madurez de las Capacidades de los Procesos de Proyectos Desarrollo; Capacidad del Proceso de Desarrollo de Software; Modelos de Madurez de las Capacidades de los procesos de Desarrollo de Software.

Capítulo III, presenta el marco metodológico, el cual da a conocer qué la naturaleza de esta investigación se enmarcada en el enfoque de ciencias del diseño y como procesamiento se seleccionó el método de ingeniería del método.

**CAPÍTULO I**

**EL PROBLEMA**

**1.1. Planteamiento del Problema**

Desde sus inicios, la disciplina de la Ingeniería de Software se ha orientado a la aplicación de un enfoque sistemático, disciplinado y cuantificable para el desarrollo, operación y mantenimiento de sistemas de software (IEEE, 1999). Este enfoque clásico, también conocido como método predictivo, considera prácticas gerenciales relativas a la planificación del proyecto, el aseguramiento de la calidad, así como también, el control y seguimiento de los mismos. Además, incluye prácticas operativas concernientes al análisis, diseño, pruebas, implantación, despliegue, entre otras.

Aunque, estas prácticas tradicionales en la industria del software han presentado un déficit respecto al incumplimiento del producto, el presupuesto y el cronograma, los resultados de estos proyectos han mejorado substancialmente desde que se inició la denominada crisis del software. La cual fue definida por Bauer en la *NATO Software Engineering Conference* en el año 1968, como un conjunto de deficiencias relacionadas a la alteración de las variables de triple restricción alcance, costo y tiempo. Así como también, a la gran cantidad de fallas incorporadas al producto de software por los desarrolladores durante el proceso de construcción, que hace imposible su utilización por el usuario en la etapa de producción.

Por otra parte, se tienen los desafíos a que deben enfrentarse los equipos de desarrollo para adaptarse a un entorno caótico, variable, dinámico y turbulento, que ha transformado la manera clásica de abordar los proyectos de software, particularmente debido a los cambios de los requisitos que surgen en las etapas tempranas y que afectan la mantenibilidad del mismo en la etapa producción, lo cual amerita un cambio de paradigma.

La agilidad es un paradigma emergente que ha llamado la atención a la comunidad científica del área de ingeniería de software y a gran parte de la industria del mismo, en la definición, planificación y ejecución de proyectos ágiles de desarrollo de software de corte pequeño y mediano. En vista que acorta los tiempos de desarrollo y reduce los costos asociados principalmente al mantenimiento, sin comprometer el alcance del proyecto en el que subyace el producto de software resultante.

En este contexto, el *Standish Group International* (SGI, 2013: p. 26), manifestó que uno de los factores de éxito de los proyectos en el año 2012, fue asumir un modelo de proceso adaptativo también conocido como método ágil. En el caso específico de los proyectos pequeños se identificaron en este reporte, un total de diez (10) puntos de éxito para la implementación del método en cuestión. El informe resalta que las tasas de éxito de pequeños y medianos proyectos de desarrollo de software en el periodo 2003-2012 que utilizan lenguajes, métodos y herramientas de última generación bajo el contexto ágil: el 46% de los proyectos fue exitoso, el 48% cuestionado y el 6% fracaso (SGI, 2013: 25).

En otros casos, el *World Quality Report* se basa en la entrevista de 1500 directores de Tecnologías de Información y Comunicación en 25 países. De acuerdo con el informe realizado en el periodo 2013-2014 (Teje y Buenen, 2013: 35), señala que el 83% de las empresas usan métodos ágiles para el desarrollo de sus aplicaciones, dado que le permiten adaptarse mejor a los cambios del entorno. Sin embargo, el 46% de las empresas encuestadas indicaron que no utilizan técnicas específicas para realizar las pruebas de las funcionalidades implementadas. La ausencia de este tipo de técnicas ha demostrado ser una preocupación para las organizaciones, lo cual es evidente por el hecho de que el 46% de los ejecutivos encuestados señaló la carencia de técnicas de pruebas adecuadas que encajen con el método de desarrollo ágil.

Los métodos ágiles toman su nombre después de que en 2001 un grupo de diecisiete (17) expertos en el desarrollo de software llegaron a un consenso de sus ideas y generaron el Manifiesto Ágil (Beck, et al., 2001), que estipula doce (12) principios que debe cumplir cada método sobre los siguientes aspectos: a) individuos e interacciones sobre procesos y herramientas; b) software funcionando sobre documentación taxativa; c) colaboración con el cliente sobre negociación contractual; y d) respuesta ante el cambio cuando se sigue un plan inicial.

Los métodos ágiles están fundamentados en prácticas basadas en el desarrollo iterativo e incremental para la especificación, desarrollo y entrega de productos de software, enfoque este imprescindible en un mundo dinámico que requiere flexibilidad ante la incertidumbre y el cambio (Sommerville, 2005). Algunos proyectos de software que enfrentan un elevado grado de incertidumbre pueden evitar pérdidas de inversión y retrabajos, utilizando técnicas aplicables de cualquier área de desempeño.

En este contexto, estos métodos se presentan como una opción atractiva, pero antagónica al método convencional que actúa en base a los principios de estabilidad y control del contexto, en vista que no se centra en la predicción, sino en la adaptabilidad. No ostenta un sistema de software perfectamente definido antes de su desarrollo, como paso previo para su construcción, sino que percibe las respuestas del cambio como una ventaja para la mejora continua del producto y la satisfacción del cliente, considerando la “…gerencia del cambio como un elemento inherente al propio proceso de desarrollo, permitiendo de esta manera, una mejor adaptación en entornos dinámicos y turbulentos” (Rodríguez, 2008: 4).

Bajo esta premisa, se plantea que la adopción del enfoque ágil en la industria software se ha incrementado en los últimos años, dado que maneja de modo efectivo la complejidad de un entorno cambiante donde prevalece la incertidumbre y la indeterminación, permitiendo un producto de software con ciertos rasgos de calidad.

El término calidad, según el ISO/IEC 25010 (ISO, 2011), es referido al conjunto de características de una entidad que se describen en las habilidades para satisfacer las necesidades establecidas e implícitas”, las cuales quizás “…no han sido establecidas, pero son necesidades reales cuando una entidad es usada en condiciones particulares” (ISO, 1998).

Es necesario enfatizar que la calidad del software está vinculada a un modelo de medición poco trivial para los profesionales de la computación; en efecto muchos elementos que intervienen durante el desarrollo del software son difíciles de medir cuantitativamente y se limitan a valoraciones cualitativas. Por tanto, asegurar la calidad es considerado uno de los procesos más relevantes y delicados del proceso de desarrollo de software.

El estándar ISO/IEC 25010 (ISO, 2011), divide la calidad bajo dos perspectivas, la vista del producto y la vista del usuario final, las cuales están conformes con la calidad del producto (considerada internamente y externamente) y la calidad en uso. La calidad de un producto de software está determinada, en muy buena medida, por la calidad del proceso usado para desarrollarlo y mantenerlo. Lo que antecede permite deducir, que la calidad del producto durante el desarrollo incide en la calidad en uso, lo cual repercute en altos índices de efectividad, rendimientos, productividad en la realización de las actividades y tareas del usuario en el ámbito organizacional-gerencial.

La gerencia de la calidad tiene como propósito “establecer un sistema gerencial que evite fallas en el ciclo de operación de [una organización]” (Crosby, 1979) y es definido por la ISO (2005: p. 19) como las “actividades coordinadas para dirigir y controlar una organización en lo relativo a la calidad”. La cual se realiza, según Juran, Medina y Ballester (1990), por medio de tres procesos, tales como: a) *Planificación de la calidad*: que contempla las actividades de desarrollo de los productos y servicios requeridos por los clientes; b) *Control de la calidad*: que evalúa el comportamiento real de la calidad, lo compara con los objetivos y actúa sobre las diferencias; y c) *Mejora de la calidad*: que busca aumentar la calidad a niveles superiores de los actuales.

En vista que uno de los objetivos de la gerencia de la calidad es “prevenir y reducir los problemas de calidad de una manera efectiva, se obtiene como beneficio la cuantificación de los costos derivados de la [misma]” (Cokins, 2006). La precisión de los “…costos de la calidad en la gerencia de las organizaciones de desarrollo, es uno de los elementos que determina el nivel de madurez de la gerencia de la calidad” (Crosby, 1979); por lo que su compresión y aplicación se constituye en prácticas sustantivas y operativas para la generación de valor y nuevas capacidades organizacionales.

En un estudio realizado por Juran, Gryma y Bingham (2005), se asoció el “concepto de capacidad al nivel de variación y los costos de la calidad mediante la planificación, el control y la mejora continua de los procesos”. Por tanto, la generación y la mejora de los procesos de desarrollo en una organización son un factor relevante para lograr los objetivos y metas, desde la perspectiva del desarrollo de software.

Según Paulk (1994), un proceso de desarrollo de software, es un “conjunto de actividades, métodos, prácticas y transformaciones para desarrollar y mantener software y productos asociados”. Por consiguiente, es percibido como un conjunto de métodos, técnicas y herramientas dirigidas a la construcción de un producto o servicio, el cual puede ser considerado como maduro en la medida que permita el logro de los objetivos propuestos y supere las metas previstas por la gerencia. A la vez que, mejora de manera continua su efectividad, rendimiento y productividad, bajo la noción de capacidades de procesos en las organizaciones.

De acuerdo con Paulk (1994), la capacidad de un proceso es concebido como “el rango de resultados esperados que se pueden obtener tras seguir un [modelo de madurez]”. Un modelo de madurez es concebido como un conjunto estructurado de elementos, entre los cuales se tiene: buenas prácticas, herramientas de medición, criterios de análisis, entre otros, que permite la precisión de las capacidades del proceso de desarrollo en una organización, la comparación con el mejor de la clase, utilización de normas internacionales en materia de software. Además, que admite la identificación de vacíos o debilidades y el establecimiento de procesos de mejora continua. Es por ello, que la madurez se percibe como el punto a través del cual un determinado proceso es explícitamente gestionado, efectivo, definido, controlado y medido.

Bajo este orden de ideas, la madurez es visualizada como un indicador del potencial de la calidad, que es enriquecido, consistente y aplicado al proceso a lo largo del ciclo de vida de los proyectos de desarrollo del software. Desde esta óptica, se plantea que la madurez implica determinar el incremento de la capacidad del proceso de desarrollo, particularmente para alcanzar la excelencia, este debe ser: definido, documentado, capacitado, soportado, mantenido, controlado, verificado, validado, medido y capaz de mejorar.

En tanto que, el incremento de la capacidad indica la evolución y el progreso del proceso conforme a un nivel de madurez concreto. El nivel de madurez se constituye como la base fundamental sobre la cual se puede ostentar un proceso de desarrollo maduro en una organización. Estos niveles contienen una escala de medición para valorar la madurez y la capacidad de los procesos de desarrollo. Los niveles facilitan a la organización la definición de prioridades en pro de las mejoras conforme a las áreas claves del proceso.

Las Áreas Claves del Proceso o *Key Process Area* (KPA), son una serie de “…actividades conexas que cuando se ejecutan… alcanzan un conjunto de metas, consideradas importantes para aumentar la capacidad del proceso” (Paulk, 1994). Mediante estas áreas claves, se obtiene un despliegue de los procesos de desarrollo de software que son efectivos, usables y aplicados de manera continua en las organizaciones.

En la medida que una organización alcanza la “…madurez de su proceso de desarrollo, ésta lo institucionaliza a través de políticas, estándares, y estructuras organizacionales” (Paulk, 1994), lo cual auspicia la conformación de una cultura corporativa y la instauración de una infraestructura tecnológica focalizada en el soporte de los métodos, prácticas y procedimientos que propendan a su excelencia.

Sin embargo, la madurez de los procesos de desarrollo de sistemas de software en las organizaciones que conforman la Industria del Software, se encuentran en un estado crítico, en vista que carecen de un modelo que precise el nivel de madurez y las capacidades generadas a través de la implementación de los métodos ágiles. En este sentido, si estas organizaciones no cuentan con un modelo de madurez para la evaluación de las capacidades no podrá precisarse si a través de estos métodos se ha alcanzado la excelencia.

Situación problemática que podría tener entre sus causas la existencia de factores relevantes que afectan el proceso de desarrollo, dentro de los que se encuentran: la resistencia al cambio y la cultura de trabajo que emergen al implementar modelos de madurez para la evaluación de las capacidades de los procesos de desarrollo que no están conformes con los principios del Manifiesto Ágil, lo cual incide en los planes de desarrollo tecnológicos institucionales, debido a la complejidad inherente a los equipos de trabajo, a la hora de emprender interacciones y vinculaciones para alcanzar un auténtico y verdadero proceso ágil de desarrollo.

Lo cual tiene como consecuencias en las organizaciones de desarrollo: aplicación de cláusulas de penalidad por incumplimiento de contratos; deterioro de la imagen institucional; alto índice de trabajos repetidos; insatisfacción y perdidas de clientes; reducción del portafolio de proyectos. Por lo cual, se hace necesario realizar una investigación que responda la siguiente interrogante general:

¿Cuáles elementos de los métodos ágiles pudieran ser utilizados para el diseñado de un modelo integrado de evaluación de la madurez de las capacidades del proceso de desarrollo de sistemas de software?

De esta interrogante, se desprenden las siguientes interrogantes específicas:

¿Qué procesos y prácticas, son desarrollados a través de los métodos ágiles de los proyectos de desarrollo de sistemas de software?

¿Qué principios del Manifiesto Ágil, pudieran ser considerados para la generación de un modelo de madurez integrado conforme a las realidades de la Industria del Software, en marco de los métodos ágiles de los proyectos de desarrollo de sistemas de software?

¿Qué modelos de madurez para la evaluación de la madurez de las capacidades del proceso de desarrollo de sistemas de software disponibles en la literatura, pudieran ser comparados para la elección de un modelo entre varios candidatos?

¿Cuáles serán las áreas claves de procesos y los indicadores para la evaluación de la madurez de las capacidades de los métodos ágiles de los proyectos de desarrollo de sistemas de software?

¿Qué directrices orientadas al diseño de un modelo integrado para la evaluación de la madurez de las capacidades deben ser definidas, conforme a los métodos ágiles de los proyectos de desarrollo de sistemas de software?

**Objetivos de la Investigación**

**Objetivo General**

Diseñar un modelo integrado para la evaluación de la madurez de las capacidades de los métodos ágiles de los proyectos de desarrollo de sistemas de software.

**Objetivos Específicos**

1. Diagnosticar los procesos y las prácticas desarrolladas a través de los métodos ágiles de los proyectos de desarrollo de sistemas de software.
2. Analizar los principios del Manifiesto Ágil en marco de los métodos ágiles de los proyectos de desarrollo en las realidades de la Industria del Software.
3. Comparar los diferentes modelos de madurez para la evaluación de la madurez de las capacidades del proceso de desarrollo de sistemas de software que se encuentran disponibles en la literatura.
4. Determinar las áreas de procesos claves y los indicadores para la evaluación de la madurez las capacidades de los métodos ágiles de los proyectos desarrollo de sistemas de software.
5. Establecer directrices orientadas al diseño de un modelo integrado para la evaluación de la madurez de las capacidades conforme a los métodos ágiles de los proyectos de desarrollo de sistemas de software.

**Justificación y relevancia de la Investigación**

La razón por la cual se justifica esta investigación reside en la necesidad que existe en la Industria del Software en determinar la madurez de las capacidades de los métodos ágil de los proyectos de desarrollo de sistemas de software, con la finalidad de contribuir un poco al cierre de la brecha entre la implementación de estos métodos y el mejoramiento continuo del proceso subyacente en el tiempo.

Este estudio contribuirá a solucionar problemas del ámbito computacional debido a la carencia de un modelo integrado adecuado para la evaluación de la madurez de las capacidades de los procesos ágiles de desarrollo, lo cual redunda en la mejora de la imagen de las organizaciones de tecnologías y el incremento de los niveles de satisfacción del usuario, logrando de esta manera la incorporación de un conjunto de indicadores y una línea base con miras de alcanzar la excelencia de las mismas. El modelo integrado propuesto generará ventajas competitivas en la Industria del Software, dado que al mejorar de manera continua el proceso de desarrollo mediante la aplicación de métodos, técnicas y herramientas de evaluación aquí propuestas, mejorará el nivel de servicios y la posibilidad de dar cumplimiento a las necesidades de los clientes, generando un valor agregado del producto.

La importancia de la presente investigación reside esencialmente en la necesidad que existe, en la Industria del Software de contar con un modelo integrado para la valoración de la madurez de las capacidades del proceso en cuestión, debido al desconocimiento de sus niveles de madurez, y por ende, de las capacidades generadas a través de la implementación de estos métodos.

En este sentido, el modelo planteado orientará a la gerencia y a los diferentes equipos de proyectos de desarrollo, por medio de una serie de medidas tecnológicas para la transformación de las debilidades en fortalezas, que facilitarán el desarrollo de planes dirigidos a generar soluciones, que cumplan las normas que manejan estas organizaciones en la industria del software.

Lo ideal es proponer un modelo de madurez para la evaluación de las capacidades de los procesos ágiles de desarrollo de sistemas de software, que guie a la gerencia en la medición y facilite un poco, el cierre de brecha entre la implantación de estos métodos y su mejoramiento continua en el tiempo, mediante la proposición de un conjunto de indicadores conforme a los niveles de madurez establecidos y la conformación de una línea base para la comparación de los resultados obtenidos de las evaluaciones previas con los resultados obtenidos en las evaluaciones actuales y posteriores, con el propósito de obtener una radiografía de la situación actual de las capacidades del proceso y su evolución histórica, que sirva de fundamento para toma de decisiones adecuadas.

La presente investigación, constituye una motivación al desarrollo de conocimientos en el contexto tecnológico, debido a que propone soluciones a partir de la aplicación del método científico a una realidad tecnológica. En el contexto personal, se plantea el reconocimiento del nivel de experticia y experiencia sobre el área de conocimiento, así como el sentimiento de colaborar con el mejoramiento de la Industria del Software, en beneficio de las sociedades mundiales que le presta sus servicios para la satisfacción de necesidades individuales y colectivas.

En el contexto académico, la presente investigación admite la pertinencia del programa de Maestría de Ingeniería del Software del Decanato de Ciencias y Tecnología de la Universidad Centroccidental “Lisandro Alvarado” (DCYT-UCLA), en vista que exterioriza el dominio y la aplicación del conocimiento suministrado al desarrollo de los cursos correspondientes. Es evidente entonces la importancia, que las organizaciones desarrolladoras de software, asuman el reto del modelo integrado que se pretender proponer, pues con ello se alcanzaría un avance considerable en el desarrollo de la Industria del Software, generando valor a sus activos, por medio del desarrollo y la innovación de productos y servicios que satisfagan las exigencias de la sociedad, proporcionándole a las organizaciones un mejor rendimiento y nuevas capacidades tecnológicas.

**Alcances de la Investigación**

En primer lugar, este trabajo se orienta hacia el diagnóstico de los procesos y prácticas desarrolladas desarrolladas a través de los métodos ágiles de los proyectos de desarrollo de sistemas de software en las organizaciones que integran la Industria del Software.

Por una parte, se pretende indagar si las organizaciones de desarrollo consideran algún método ágil de desarrollo como: AUP, Scrum o XP. Es necesario enfatizar, que cada método cuenta con un macroproceso que se constituye en el ciclo de vida de proyectos, denominado como “conjunto de fases que entrecruzan a un proyecto desde su inicio hasta su cierre” (PMI, 2017: 547). Cada fase es a su vez un “conjunto de actividades de un proyecto, relacionadas de manera lógica, que culmina con la finalización de uno o más entregables” (ídem), pueden ser secuenciales, iterativas o superpuestas.

Y, por otra parte, se pretende determinar las prácticas utilizadas por estas organizaciones en el desarrollo de proyectos de sistemas software. Es necesario enfatizar, que una práctica es la aplicación operativa de la teoría que subyace a un método ágil en particular, se encamina al conocimiento práctico o experiencial, el cual se dirige hacia la obtención de la efectividad y la generación de valor. En cambio, la teoría es la construcción teórica per se acerca de un área específica, ambas categorías coexisten de manera dialéctica: teoría-praxis. Las prácticas que interesan en el presente trabajo son aquellas desarrolladas en las organizaciones en la definición de sus proyectos, entre las cuales, se tiene prácticas gerenciales y las prácticas operativas.

En segundo lugar, este trabajo se dirige hacia el análisis de los principios del Manifiesto Ágil en marco de los métodos ágiles de desarrollo en las realidades de la Industria del Software, que se encuentran disponibles en la literatura con el objeto de indagar cuáles principios aplican a los métodos ágiles de desarrollo de sistemas de software.

En tercer lugar, el presente estudio compara los diferentes modelos de madurez para la evaluación de la madurez de las capacidades del proceso de desarrollo de sistemas de software que se encuentran disponibles en la literatura.

En cuarto lugar, esta investigación se centra en la precisión de las áreas de procesos claves y los indicadores para la evaluación de la madurez las capacidades de los métodos ágiles de desarrollo de sistemas de software.

En quinto lugar, el presente estudio está dirigido a la definición de directrices orientadas al diseño de un modelo integrado para la evaluación de la madurez de las capacidades conforme a los métodos ágiles de los proyectos de desarrollo de sistemas de software.

Y finalmente, en sexto lugar, los resultados obtenidos de la definición de las directrices serán considerados para el diseño un modelo integrado para la evaluación de la madurez de las capacidades de los métodos ágiles de desarrollo de sistemas de software.

**CAPÍTULO II**

**MARCO TEÓRICO**

En esta sección se exponen los antecedentes de la investigación, dentro de los cuales se encuentran trabajos de grado, artículos arbitrados y tesis doctorales que de alguna manera conllevaron a la realización de la propuesta de solución al problema planteado. Es necesario señalar que existen pocas investigaciones en torno al tema de investigación, por tanto, se seleccionaron aquellos trabajos que guardan alguna relación con el problema objeto de estudio:

**Antecedentes de la Investigación**

Alonso (2017), en su trabajo de grado de maestría titulado ***Modelo CMMI y métodos agiles en la gestión de proyectos de software***. Maestría interuniversitaria en Dirección de Proyectos, Departamento de Explotación y Prospección de Minas, Universidad de Oviedo, Oviedo, España. El objetivo general consistió en

Concluye: a) Combinar CMMI y los métodos ágiles, puede ayudar a que las organizaciones basadas en desarrollo sean capaces de responder más rápidamente a cambios en su entorno. b) Los grandes proyectos software, se pueden beneficiar de la aplicación de los métodos agiles, ya que suelen generar sistemas más modulares, fáciles de mantener y evolucionar. c) CMMI y los métodos agiles si se utilizan correctamente pueden ayudar a las organizaciones a que sus proyectos alcancen un equilibrio óptimo. d) El modelo propuesto ayuda a que los procesos sean controlados, gestionados, medidos y analizados de forma sistemática e institucionalizada. Y e) la agilidad permite reducir trabas burocráticas, formalismos innecesarios y otros impedimentos que lastran el rendimiento del proyecto.

Entre los aportes que proporcionó este trabajo a la presente investigación, se tiene: a) Evaluar el nivel de madurez y las capacidades del uso de otros métodos métodos ágiles tales como ASD, AUP, XP y DSDM. Y b) un modelo de madurez para la evaluación de las capacidades de un proceso ágil de desarrollo de software de basarse en CMMI dado que integra CMM y el ISO/IEC 15504.

Rodríguez (2017), en su trabajo de grado de maestría titulado ***Análisis y diseño de un modelo con integración de una metodología ágil en el nivel 2 de CMMI***. Maestría en Ingeniería – Telecomunicaciones, Facultad de Ingeniería, Universidad Nacional de Colombia, Bogotá, Colombia. El objetivo general consistió en generar modelo híbrido que se nutre de la definición de las áreas estratégicas y procesos de madurez provistas por el modelo CMMI DEV y las estrategias de aplicación específicas provistas por la metodología de desarrollo de software ágil Scrum.

Concluye: a) En Colombia aproximadamente el 60% de las empresas de tecnologías no cuentan con marcos de referencia de calidad avalados internacionalmente. b) La implementación de modelos de madurez como CMMI tardan entre 12 y 18 meses. c) Es indispensable hacer uso de metodologías ágiles para minimizar el tiempo de la adopción de los diferentes modelos de calidad. d) Scrum proporciona la metodología para implementar el proceso de CMMI-DEV en áreas relacionadas con el nivel de madurez gestionado y deja la base sólida para implementar el nivel definido. Y e) la Integración de modelos de madurez de capacidades se ve influenciado por las metodologías ágiles en particular la de Scrum.

Entre los aportes que proporcionó este trabajo a la presente investigación, se tiene: a) La implementación de modelos de madurez requiere para el logro de sus objetivos requiere del apoyo de la alta gerencia. b) Relación de CMMI con Scrum. Y c) Identificación de las Áreas claves de procesos, objetivos y prácticas específicas.

Iglesias al et. (2015), en su trabajo titulado ***Modelo aplicable a la gestión de procesos ágiles de desarrollo de software basado en CMMI–DEV 1.3 y scrum***. Universidad Simón Bolívar, Barranquilla, Colombia. El objetivo general consistió en formular un modelo de gestión de procesos de desarrollo basado en los principios y prácticas del enfoque ágil, orientado hacia la aplicación de buenas prácticas y procesos de mejora continua.

Concluye: a) En la comparación de las metodologías, modelos y marcos de trabajo de los diferentes métodos ágiles, Scrum cumple con la mayoría de los criterios establecidos. b) En la revisión estadística del uso de métodos ágiles, Scrum resultó ser de las más usadas y conocidas dentro del enfoque ágil. c) En la revisión del proceso de desarrollo de Scrum se vislumbró algunas falencias en cuanto al empleo de buenas prácticas orientadas a la calidad del proceso propias del marco de trabajo, es por ello se incorporan las buenas prácticas de CMMI – DEV 1.3. d) Se realizó una correspondencia entre las prácticas genéricas y específicas de las áreas del proceso de CMMI – DEV y las metas, artefactos y objetivos del proceso de Scrum. e) se encontró que en los niveles superiores de CMMI existe una carencia de prácticas asociadas a Scrum. Y f) existen áreas del proceso de desarrollo que podrían emplearse y que de hecho son necesarias, sin sacrificar la agilidad, estas áreas serian la gestión de la configuración, la gestión de riesgos y la innovación e implementación organizacionales.

Entre los aportes que proporcionó este trabajo a la presente investigación, se tiene: a) Definir metas, objetivos, prácticas e indicadores para los niveles superiores de CMMI, procurando no sacrificar la agilidad. b) Incorporar áreas de gestión de la configuración, gestión de riesgos e implementación.

Rodríguez (2014), en su trabajo de grado de maestría titulado ***Proceso de Desarrollo de Software basado en metodologías ágiles y CMMI-DEV***. Maestría Universitario en Investigación en Ingeniería de Software y Sistemas Informáticos, UNED, Madrid, España. El objetivo general consistió en desarrollar una guía para implantar una metodología de desarrollo ágil que cumpla las buenas prácticas propuestas por el nivel 2 de madurez de CMMI, utilizando herramientas de Microsoft para su ejecución y potenciando el teletrabajo.

Concluye: a) Una buena cantidad de las prácticas específicas requeridas por CMMI nivel 2 de madurez son cumplidas sin necesidad de realizar cambio alguno a la metodología de desarrollo Scrum. b) Pocos resultados se obtuvieron de la prueba que se realizó utilizando esta metodología en el proyecto piloto donde se realizaron tres (3) Sprints. Y c) fue irrelevante el tiempo invertido en documentación.

Entre los aportes que proporcionó este trabajo a la presente investigación, se tiene: a) Identificar en el proceso Scrum las actividades que permitían cumplir con los requerimientos de CMMI-Dev 2. Y b) es necesario realizar adecuaciones en el método para que cumpla con las metas y prácticas genéricas establecidas por CMMI-DEV.

Iribarren (2012), en su trabajo de grado de maestría titulado ***Modelo de Mejora Continua para la Gestión de Requisitos de Software usando Métodos Ágiles***. Maestría en Ciencias de la Computación, Decanato de Ciencias y Tecnología, UCLA, Barquisimeto, Venezuela. El objetivo general consistió en diseñar un modelo de mejora continua para la gestión de requisitos de software usando métodos ágiles.

Concluye: a) El método Scrum parte del hecho de que los requisitos van a estar cambiando a lo largo de todo el ciclo de vida de desarrollo de software. b) Los requisitos en Scrum son usados en cada una de sus etapas y en cada una de sus actividades de su ciclo de vida. c) El uso de los métodos de software para el proceso de desarrollo de software es un elemento importante para conseguir la satisfacción de los clientes que requieren el producto de software. Y d) el uso del Diagrama de Proceso de Negocio en BPMN, permite a los responsables de los procesos conocer de una manera clara y estandarizada con una notación sencilla los procesos que se llevan en la empresa y permiten su monitoreo, control y automatización.

Entre los aportes que proporcionó este trabajo a la presente investigación, se tiene: a) Uso de una herramienta basada SPEM 2.0 para el modelado del proceso ágil de desarrollo de software. Y b) utilización de indicadores para la evaluación del desempeño de los diferentes procesos.

**Bases Teóricas**

**Modelos de Proceso de Desarrollo de Sistemas de Software**

En principio, un proceso es una actividad especial que describe una estructura de tipos particulares de proyectos de desarrollo o partes de estos. Según la OMG (2008: 174), un “…proceso puede ser adaptado o instanciado a una organización o dominio específico”. Un proceso es instanciado cuando se le agrega o remueven recursos concretos, generando múltiples instancias de productos de trabajo.

El proceso de desarrollo, denominado también macroproceso, define “…cómo se desarrollan y ejecutan los proyectos,… se alcanzan los hitos para definir las subsiguientes secuencias de trabajos, operaciones o eventos que usualmente requieren tiempo, experiencia y recursos, para producir algún resultado” (OMG, 2008: 156), contempla un conjunto de fases que expresan el ciclo de vida de un producto o servicio en desarrollo, ostentan un inicio y un cierre.

Según Paulk (1994), un proceso de desarrollo de software es un “conjunto de actividades, métodos, prácticas y transformaciones para desarrollar y mantener software y productos asociados”. Por consiguiente, es percibido como un conjunto de métodos, técnicas y herramientas dirigidas a la construcción de un producto o servicio, el cual puede ser considerado como maduro en la medida que permita el logro de los objetivos propuestos y supere las metas previstas por la gerencia. A la vez que, mejora de manera continua su efectividad, rendimiento y productividad, que en el mundo computacional es asociado a la noción de modelos.

Un modelo de procesos de proyectos se constituye en el ciclo de vida, denominado “conjunto de fases que entrecruzan a un proyecto desde su inicio hasta su cierre” (PMI, 2017: 547), que representa la evolución de un proyecto. Cada fase es a su vez un “conjunto de actividades de un proyecto, relacionadas de manera lógica, que culmina con la finalización de uno o más entregables” (Ídem). Según el PMI (2017), los modelos de ciclos de vida son:

* *Predictivo*: alcance, tiempo y costo del proyecto se determinan en las fases tempranas del ciclo de vida. Cualquier cambio en el alcance se gestiona cuidadosamente.
* *Iterativo*: el alcance del proyecto generalmente se determina tempranamente en el ciclo de vida del proyecto, pero las estimaciones de tiempo y costo se modifican periódicamente conforme aumenta la comprensión del producto por parte del equipo del proyecto. Las iteraciones desarrollan el producto a través de una serie de ciclos repetidos, mientras que los incrementos van añadiendo sucesivamente funcionalidad al producto.
* *Incremental*: el entregable se produce a través de una serie de iteraciones que sucesivamente añaden funcionalidad dentro de un marco de tiempo predeterminado. El entregable contiene la capacidad necesaria y suficiente para considerarse completo sólo después de la iteración final.
* *Adaptativos* (ágiles, iterativos o incrementales): El alcance detallado se define y se aprueba antes del comienzo de una iteración. Los ciclos de vida adaptativos también se denominan ciclos de vida ágiles u orientados al cambio.
* *Híbrido*: es una combinación de un ciclo de vida predictivo y uno adaptativo. Aquellos elementos del proyecto que son bien conocidos o tienen requisitos fijos siguen un ciclo de vida predictivo del desarrollo, y aquellos elementos que aún están evolucionando siguen un ciclo de vida adaptativo del desarrollo (19).

De lo anterior, se deduce que un modelo de procesos de proyectos es una plantilla, patrón o marco que define el proceso a través del cual se crea un producto o servicio. Por ende, los procesos son instancias de un modelo de proceso. Las organizaciones pueden adaptar su modelo de procesos en correspondencia a la naturaleza del proyecto conforme al producto o servicio, los métodos y herramientas a utilizar, los controles y entregas requeridas.

**Enfoque ágil de los proyectos de desarrollo de Sistemas de Software**

En principio, Owen et al. (2006), definen la agilidad se define como la “…cualidad de actuar de manera proactiva en un entorno dinámico, turbulento y en constante cambio”. De esta aseveración, se plantea que el concepto de agilidad se inició en el dominio del desarrollo de software para abordar la naturaleza volátil de los productos o servicios de software, la incertidumbre y la dificultad de la definición de los requisitos en las etapas tempranas del proceso de desarrollo.

Bajo esta óptica, una característica única del desarrollo ágil es que “cada iteración es autónoma con respecto a las actividades que van desde el análisis de los requisitos hasta llegar a la implementación” (Larman, 2004). Al final de cada iteración, es presentada a cliente una versión que integra todos los componentes del software. El cliente proporciona la retroalimentación y señala las mejoras necesarias en los requisitos y las características del sistema, que deben ser planificadas y consideradas en las versiones o iteraciones futuras. El desarrollo de software ágil es impulsado por el principio de entrega de dirigida por valor para satisfacer las necesidades de los clientes a través de la entrega temprana y continua, en correspondencia a las prioridades de las características del producto o servicio de software.

Por otra parte, la agilidad organizacional es considerada como “la capacidad de una organización para adaptarse a nuevas condiciones sin verse obligados al cambio” (Ali, Chew y Tang, 2004).

En este contexto, los métodos ágiles de proyectos de desarrollo son vistos como una combinación de los conceptos de los métodos de desarrollo tradicionales y las prácticas flexibles, ligeras, colaborativas y adaptables a los cambios frecuentes del entorno, pero altamente disciplinadas. La noción de enfoque ágil de desarrollo ha sido altamente influenciada por los conceptos de los métodos de desarrollo de software ágiles, tales como: Scrum, eXtreme Programming (XP), Agile Unified Process (AUP) los cuales están orientados a un conjunto de principios de desarrollo ágil. Por ende, se plantea que sea un enfoque basado en principios de desarrollo ágil que guían los roles, las relaciones y las actividades del proceso con el equipo, la gerencia y los clientes de un proyecto concreto.

Es un enfoque dinámico en vista que permite la modificación continua de un proyecto, en tanto que los requisitos son revisados y evaluados en cada iteración. Por otra parte, sigue un enfoque de gerencia de dirigido por características; se centra en la definición del alcance y los requisitos del proyecto, dando prioridad a la lista tanto de características como de necesidades basadas en la agregación de valor, así como en la maximización de los ingresos o cuota del mercado.

En este enfoque, es crucial la participación del cliente en el alcance y el análisis de los requisitos del proyecto. Esto garantiza que el equipo de proyecto ágil no invierta esfuerzo en el trabajo de bajo valor agregado o en características o requisitos costosos y sin sentido. De acuerdo con Hass (2007), enfatiza en el “…desarrollo colaborativo y en la gestión para obtener resultados, retroalimentación de los clientes y mejoramiento continuo”.

Finalmente, se plantea que este enfoque se fundamente en modelos de procesos iterativos e incrementales, donde los miembros del equipo y los stakeholders (partes interesadas) colaboran activamente para comprender el dominio del proyecto, identificar lo que tiene que ser construido y establecer las prioridades de las funcionalidades requeridas.

**Principios del Manifiesto Ágil de Desarrollo de Sistemas de Software**

Según Beck al et. (2001), el Manifiesto Ágil del Desarrollo de Software surgió en febrero 2001 y estipula doce (12) principios que deben seguir los métodos ágiles de desarrollo de software, los cuales se muestran a continuación:

1. Nuestra mayor prioridad es satisfacer al cliente mediante la entrega temprana y continua de software con valor.
2. Aceptamos que los requisitos cambien, incluso en etapas tardías del desarrollo. Los procesos Ágiles aprovechan el cambio para proporcionar ventaja competitiva al cliente.
3. Entregamos software funcional frecuentemente, entre dos semanas y dos meses, con preferencia al periodo de tiempo más corto posible.
4. Los responsables de negocio y los desarrolladores trabajamos juntos de forma cotidiana durante todo el proyecto.
5. Los proyectos se desarrollan en torno a individuos motivados. Hay que darles el entorno y el apoyo que necesitan, y confiarles la ejecución del trabajo.
6. El método más eficiente y efectivo de comunicar información al equipo de desarrollo y entre sus miembros es la conversación cara a cara.
7. El software funcionando es la medida principal de progreso.
8. Los procesos Ágiles promueven el desarrollo sostenible. Los promotores, desarrolladores y usuarios debemos ser capaces de mantener un ritmo constante de forma indefinida.
9. La atención continua a la excelencia técnica y al buen diseño mejora la Agilidad.
10. La simplicidad, o el arte de maximizar la cantidad de trabajo no realizado, es esencial.
11. Las mejores arquitecturas, requisitos y diseños emergen de equipos autoorganizados.
12. A intervalos regulares el equipo reflexiona sobre cómo ser más efectivo para a continuación ajustar y perfeccionar su comportamiento en consecuencia.

Estos principios serán considerados en el presente estudio para la conformación de la propuesta.

**Noción disciplinaria de los equipos en el proceso de desarrollo de sistemas de software**

Para Morin (1998, p.1) una disciplina es una “…categoría organizacional en el seno del conocimiento científico; la cual constituye una división y especialización del trabajo y responde a la diversidad de dominios que recubren las ciencias”. Una disciplina se engloba en un conjunto de conocimientos científicos, los cuales tienden “….naturalmente a la autonomía, por la delimitación de sus fronteras, el lenguaje que se establece, las técnicas que conduce para su elaboración o utilización, y eventualmente por teorías que le son propias” (Morin, 1998, p.1). Desde el punto de vista epistemológico existen tres (3) enfoques disciplinarios, tales como:

1. *Multidisciplinariedad*: desde el punto de vista etimológico significa “…muchas disciplinas, es decir, varias disciplinas abordando el mismo objeto de estudio pero sin interconexión alguna o relación aparente entre ellas” (Carmona, 2004: 2). Para abordar cualquier dominio de conocimiento se utilizan múltiples disciplinas que convergen en la resolución de un determinado problema. La multidisciplinariedad hace referencia a “…distintas disciplinas, a la división de los campos científicos, al desarrollo y necesidades de las ramas del saber, a lo más específico y propio del desarrollo científico-técnico y a la profundización de los conocimientos” (Boiero y Estrada, 2008: 2). Bajo esta concepción, se plantea que en este enfoque las disciplinas, ciencias o dominios están orientados y dan razón del conocimiento sobre la particularidad de un problema.

2. *Interdisciplinariedad*: Tamayo y Tamayo (2004: 172) la plantean como un enfoque que posee un “…conjunto de disciplinas conexas entre sí y con relaciones definidas, a fin de que sus actividades no se produzcan”. Bajo esta afirmación, se plantea que en el citado enfoque exista una sinergia entre varias disciplinas. Esta integración o intercambio entre disciplinas es de “…naturaleza intrínseca de sus métodos, o sea, al momento de abordar un problema u objeto de estudio sus métodos son comunes, de una u otra área del conocimiento” (Carmona, 2004: 2). De allí que este enfoque requiera y promueva la colaboración y cooperación de diversas disciplinas científicas y especialistas procedentes de diferentes dominios de conocimientos de manera aislada, dispersa y fraccionada para solucionar un problema determinado.

3. *Transdisciplinariedad*: según (Nicolescu, 1997: 6) el prefijo *trans* indica que “… es a la vez entre las disciplinas, a través de las diferentes disciplinas, y más allá de toda disciplina. Su propósito es la comprensión del mundo actual, uno de los imperativos [de este enfoque] es la unificación del conocimiento”. De allí que la transdisciplinariedad sea concebida como la resolución de problemas entre, a través de y más allá de las disciplinas. La transdisciplinariedad implica interdependencia de los niveles de la ciencia, lo cual permite trascender las fronteras para lograr, según Martínez (2007: 2), “…una genuina integración entre las disciplinas, de tal modo que se consiga la sinergia”, para la resolución de problemas. Excluyendo de esta manera la verticalidad de las investigaciones como parte del proceso. Cabe señalar que dentro del enfoque transdiciplinario se encuentran “…insertados otros conceptos de autoorganización, auto mantenimiento, autotransformación, autorrenovación y auto transferencia, todos los cuales configuran una especie de autopoiesis, es decir, una especie de auto creación (Martínez, 2007: 2).

Los citados enfoques disciplinarios conllevan a la conformación de equipos multi, inter y transdisciplinarios, los cuales se describen a continuación (Martínez, 2003):

* *Equipo multidisciplinario*: los participantes colaboran en un proyecto común. Pero pertenecen a diversas disciplinas y cada uno es básicamente independiente en su trabajo, sintiendo poca o ninguna necesidad de conocer el trabajo de los demás. (13).
* *Equipo interdisciplinario*: los participantes pertenecen a diferentes disciplinas, pero la integración comienza en el mismo proceso, en la formulación del plan de acción y en la especificación de la contribución de cada miembro: cada uno trata de tener en cuenta los procedimientos y el trabajo de los otros en vista a una meta común que define el proyecto. Por ello, la coordinación, la comunicación, el diálogo y el intercambio son esenciales, para traducir términos propios, aclarar lenguajes ambiguos, seguir, aunque sea parcialmente, procedimientos metodológicos similares, y en general, tratar de compartir algunos de los presupuestos, puntos de vista y lenguajes de otros (15).
* *Equipo transdisciplinario*: los participantes transcienden sus propias disciplinas (o las ven sólo como complementarias) logrando crear un nuevo mapa cognitivo común sobre el problema en cuestión, es decir, llegan a compartir un marco epistémico amplio y una cierta meta-metodología que les sirven para integrar conceptualmente las diferentes orientaciones de sus análisis: postulados o principios básicos, perspectivas o enfoques, procesos metodológicos, instrumentos conceptuales, etc. (17).

De lo que antecede se infiere que: a) En los equipos multidisciplinarios: la integración de los resultados se realiza al final del proceso de construcción; b) En los equipos interdisciplinarios: existe una auténtica integración tanto conceptual como de los resultados del proceso, utilizando normas y estándares preestablecidos; Y c) en los equipos transdisciplinarios: el participante posee un marco conceptual de referencia y un proceso unificado que le permite obtener un resultado integral.

**Métodos Ágiles de Proyectos de Desarrollo de Sistemas de Software**

A continuación, se presentan brevemente los métodos de gerencia de proyectos ágiles considerados en el presente estudio:

***Agile Unified Process (AUP) o Proceso Unificado Ágil***

Iniciado por Jacobson, Booch y Rumbaugh (1999, p. 40), define “quién está haciendo: el equipo de trabajo; qué están haciendo: en referencia a los artefactos; cuándo debe hacerlo: en relación a las fases del proceso; y cómo obtener un cierto objetivo: en correspondencia al encadenamiento de actividades”. De cual se infiere que el referido método está enmarcado en un ciclo de vida orientado hacia las dimensiones problema-solución que permite reducir riesgos en el desarrollo de un proyecto concreto.

Según Firestone (2001, p. 61), el Proceso Unificado esta “…divido en fases que a su vez se subdividen en iteraciones o mini-proyectos, que producen un incremento de la solución hasta alcanzar los objetivos planteados en un proyecto particular”. Está constituido por cuatro (4) fases, tales como: “*Inicio*: Define el alcance y factibilidad del proyecto. *Elaboración*: Planifica el proyecto, especifica las características y la arquitectura base. *Construcción*: desarrollo del producto. Y *Transición*: Entrega del producto al cliente o a los usuarios y mantenimiento” (Larman 2002, p. 60).

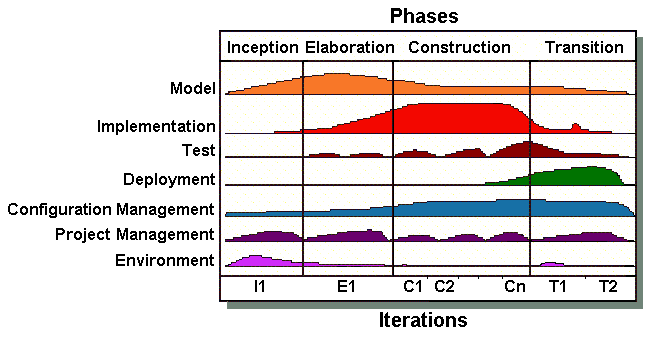
Una de las cualidades más resaltantes del Proceso Unificado, es que es un método configurable y/o extensible, debido a que fue desarrollado de manera tal que soporta cambios que no alteran su estructura de diseño, permitiendo agregar y remover disciplinas o flujos de trabajo existentes. De allí que el referido modelo pueda ser adaptado o personalizado a las necesidades de una organización o a un dominio específico.

El Proceso Unificado Ágil o *Agile Unified Process* (AUP) de Scott Ambler (2006), es una versión o instanciación simplificada del Proceso Unificado. Este describe de una manera simple y fácil de entender la forma de desarrollar aplicaciones de software del negocio usando técnicas ágiles y conceptos que aún se mantienen válidos en del Proceso Unificado. Aplica técnicas ágiles incluyendo Desarrollo Dirigido por Pruebas (*Test Driven Development* - TDD), Modelado Ágil, Gestión de Cambios Ágiles y Refactorización para mejorar la productividad. Se basa en los siguientes principios (Ambler, 2006):

* *Equipo de desarrollo*: requiere directrices alto nivel y/o la formación. El enfoque suministra muchas técnicas, pero se recomienda el uso de las que se adapten a las necesidades del proyecto.
* *Simplicidad*: Todo se especifica de manera concisa y sencilla.
* *Agilidad*: Se ajuste a los valores y principios del Manifiesto Ágil.
* *Focalizado en actividades de agregación de valor*: La atención se centra en las actividades que realmente cuentan, no todo lo posible que podría suceder a usted en un proyecto.
* *Independencia de herramientas*: Se puede utilizar cualquier conjunto de herramientas que se desee. Se sugiere que se utilicen herramientas adecuados al trabajo.
* *Adaptabilidad*: Se adapta fácilmente a las necesidades del equipo y al contexto del proyecto.

Es necesario señalar que el modelo de proceso de AUP ostenta las mismas fases del Proceso Unificado. Sin embargo, las perspectivas las cuales contienen las diferentes disciplinas del proceso, las cuales se llevan a cabo de manera sistemática, en ellas se definen las actividades que realizan los miembros del equipo de desarrollo multidisciplinario para construir, validar y desplegar software de trabajo que responda a las necesidades de sus grupos de interés. Las disciplinas de AUP fueron adaptadas al contexto de la Gerencia de Proyectos Ágiles (Ambler, 2006), entre las cuales se tiene (ver figura 1):

* *Modelo*: El objetivo de esta disciplina es entender el negocio de la organización, el dominio del problema siendo abordado por el proyecto, y para identificar una solución viable para hacer frente al dominio del problema.
* *Implementación*: El objetivo de esta disciplina es transformar su modelo (s) en código ejecutable y para llevar a cabo un nivel básico de las pruebas, en las pruebas de unidad en particular.

****

**Figura 1. Método AUP**

**Fuente**: adaptado de Scwaber (1995)

* *Pruebas (Test)*: El objetivo de esta disciplina es llevar a cabo una evaluación objetiva para garantizar la calidad. Esto incluye encontrar defectos, validar que el sistema funciona tal como fue diseñado, y verificar que se cumplen los requisitos.
* *Despliegue*: El objetivo de esta disciplina es planificar para la entrega del sistema y ejecutar el plan para que el sistema disponible para los usuarios finales.
* *Gestión de configuraciones*: El objetivo de esta disciplina es para administrar el acceso a sus artefactos del proyecto. Esto incluye no sólo el seguimiento de versiones de artefactos en el tiempo, sino también el control y la gestión de los cambios a los mismos.
* *Gerencia de proyectos*: El objetivo de esta disciplina es dirigir las actividades que lleva a cabo el proyecto. Esto incluye los riesgos de gestión y dirección de personas (la asignación de tareas, seguimiento del progreso, etc.), y la coordinación con las personas y los sistemas fuera del alcance del proyecto para asegurarse de que está entregado a tiempo y dentro del presupuesto.
* *Ambiente*: El objetivo de esta disciplina es apoyar el resto del esfuerzo por garantizar que el proceso adecuado, orientación (normas y directrices) y herramientas (hardware, software, etc.) están disponibles para el equipo, según sea necesario.

**Scrum**

Se basa en proceso empírico, iterativo e incremental de desarrollo que intenta obtener ventajas respecto a los procesos definidos (cascada, espiral, prototipos, etc.) mediante la aceptación de la naturaleza compleja del desarrollo de software, y la utilización de prácticas tendientes a manejar el caos, la impredictibilidad y el riesgo a niveles aceptables. El mismo surge de un estudio presentado por Takeuchi y Nonaka en 1986 en el *Harvard Business Review* titulado “*The New Product Development Game*”, que propone las mejores prácticas más utilizadas en las diez (10) compañías japonesas altamente innovadoras. A partir de allí y tomando como referencia el juego de *rugby*, Jeff Sutherland y Ken Scwaber en año 1995 (Sutherland y Schwaber, 1995), formalizan el método Scrum (ver figura 2).

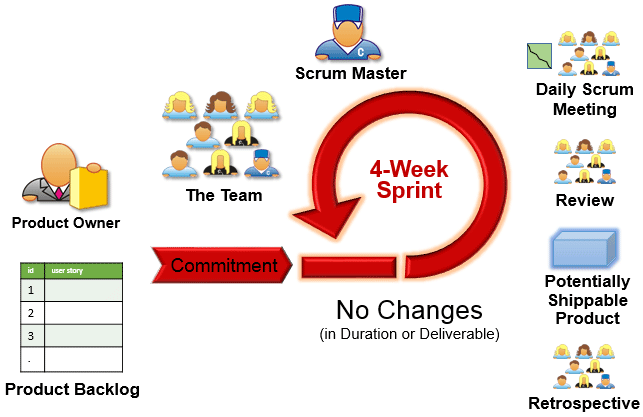


Figura 2. **Método Scrum**

**Fuente**: adaptado de Scwaber (1995)

Es un método iterativo e incremental que se enfatiza en prácticas y valores de la gerencia de proyectos, sobre las demás disciplinas de construcción de software. Al principio del proyecto se define el *Product Backlog*, que contiene las historias de usuarios (requisitos funcionales). Es necesario enfatizar que en la definición de esas historias de usuarios está conformada por el Actor, la Acción y el Objetivo. Los objetivos o metas funcionales se hacen corresponder con las metas no funcionales que deberá satisfacer el sistema objeto de desarrollo.

Las historias de usuarios son especificadas de acuerdo a las convenciones de la organización bien sea mediante: features, casos de uso, diagramas de flujo de datos, tareas, etc. El *Product Backlog* es definido durante reuniones de planificación con los *stakeholders*. De allí en adelante, se definen las iteraciones, conocidas como *Sprint* en el juego del Scrum, en las que irá incrementando evolutivamente la aplicación. Cada *Sprint* tiene su propio *Sprint Backlog* que es un subconjunto del *Product Backlog* con las historias de usuarios a ser desarrolladas en el *Sprint* correspondiente. La duración recomendada del *Sprint* es de 1 a 4 semanas.

Por otra parte, en cada *Sprint* el *Scrum Master* (equivalente al Gerente de Proyectos) lleva a cabo la gestión de la iteración, convocando diariamente al *Scrum Daily Meeting* que representa una reunión de avance diaria de no más de 15 minutos con el propósito de tener realimentación sobre las tareas de los recursos y los obstáculos que se presentan. Al final de cada *Sprint*, se realiza un *Sprint Review* para evaluar los artefactos construidos y un *Sprint Retrospective* para dar a conocer los problemas suscitados y las soluciones que serán implementadas por el *Equipo Scrum*. Así mismo, un día antes de la planificación se realiza el *pre-sprint* o *pre-planificación* del *Sprint* subsiguiente. En la planificación del Sprint el (o los) equipo(s) selecciona(n) las historias de usuarios *INVEST* que desean desarrollar.

La intención de *Scrum* es la de maximizar la realimentación sobre el desarrollo a través de la resolución problemas y la mitigación de riesgos de manera temprana. Su uso se está extendiendo cada vez más dentro de la comunidad de Metodologías Ágiles, siendo combinado con otros enfoque como *eXtreme Programming* (o Programación Extrema) para complementar sus carencias. Existe una tendencia en la Industria a nivel mundial en la incorporación y al uso de Scrum en el desarrollo e innovación de productos y servicios tecnológicos.

***eXtreming Programing* (XP) o Programación Extrema**

La programación extrema o eXtreme Programming, iniciada Kent Beck (1999), se basa en una serie de reglas y principios que se han ido gestando a lo largo de toda la historia de la gerencia de proyectos de software. Usadas conjuntamente proporcionan un proceso de desarrollo de proyectos que se puede englobar dentro de los métodos ligeros, que son aquéllas en la que se da prioridad a las tareas que dan resultados directos y que reducen la burocracia que hay alrededor tanto como sea posible. La programación extrema, dentro de los procesos ágiles, se puede clasificar dentro de los *evolutivos*.

Una de las características de XP es que muchos de, si no todos, sus elementos son conocidos dentro de la rama de la gerencia de proyectos desde hace tiempo, incluso desde sus comienzos. Los autores han seleccionado los que han considerados como los mejores y han profundizado en sus relaciones y en cómo se refuerzan unos a otros. El resultado ha sido un método único y compacto (ver figura 3). Por eso, aunque se pueda alegar que la programación extrema no se base en principios nada nuevos, se ha de aclarar que, en conjunto, es una nueva forma de ver el desarrollo de proyectos tecnológicos.

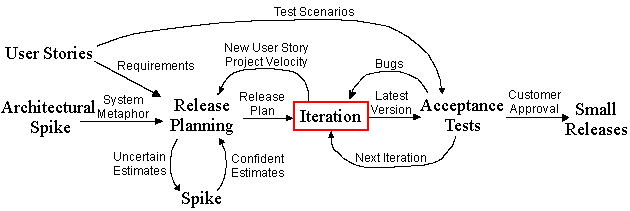


Figura 3. **Método XP**

**Fuente**: adaptado de Beck (1999)

Aunque, como ya se ha comentado, la programación extrema se basa en valores, principios fundamentales y prácticas, en este artículo no se van a enumerar, ya que el autor considera que no es la mejor forma de presentarlos. Los principios y prácticas no se han hecho a priori, sino que tienen un por qué a partir de una forma global de desarrollo de proyectos que, al menos en teoría, parece ser más eficiente.

Por lo tanto, este método se presentará desde un punto de vista práctico para luego enunciar valores, principios y prácticas que hacen que se lleven a buen fin. Es por ello, que en la figura 3, se muestra un proceso de desarrollo extremo tal y como debería darse en un equipo de desarrollo que sigue este método. De esta forma se irán detallando y explicando las diferentes técnicas utilizadas, así como su razón de ser. El proceso se basa en cinco (5) principios, según Beck (1999):

* *Comunicación*: se basa en una cultura de la comunicación oral y sus prácticas están diseñadas para fomentar la interacción. El valor de la comunicación se basa en la observación de que la mayoría de las dificultades del proyecto se deben a respuesta a preguntas, colaboración o la obtención de ayuda.
* *Simplicidad*: Diseño del producto simple que satisfaga las necesidades del cliente. Un aspecto importante del valor es que sólo el diseño y el código está dirigido a los requisitos actuales en lugar de anticipar y planificar requisitos no declarados.
* *Feedback*: El equipo de desarrollo obtiene retroalimentación de los clientes al final de cada iteración y release. Esta retroalimentación impulsa la siguiente iteración.
* *Coraje*: Es que el equipo pueda tener valor en sus acciones y toma de decisiones.
* *Respeto*: Los miembros del equipo deben preocuparse por el compañero y por el proyecto.

Beck (1999), plantea doce (12) prácticas que sirven como un punto de partida para un equipo XP, las cuales se basan en los principios que anteceden en tres capas:

* ***Prácticas de Programación***:

1. *Diseño simple*: El diseño se va creando en forma progresiva, sin prever las necesidades del futuro. Al tener un diseño simple capaz de mantener las características actuales del sistema, este puede adaptarse mejor a un entorno cambiante cuando surgen nuevos requerimientos o cambian los actuales.
2. *Refactoring*: Significa mejorar el diseño del código sin cambiar la funcionalidad actual. Aplicar esta técnica permite reducir la complejidad del código y eliminar posibles redundancias.
3. *Estándares de código*: El equipo de programadores define en forma consensuada los estándares de programación que utilizará. De esta manera se logra obtener un código uniforme, como si el sistema fuera programado por una sola persona. Esta práctica fomenta la comunicación y facilita la implementación de las prácticas de refactoring, programación en pareja y propiedad colectiva.
4. *Pruebas*: Las pruebas (testing) son una de las prácticas fundamentales en las cuales se basa XP. Esta actividad se realiza en forma continua a lo largo del proyecto. Existen dos tipos de pruebas, las unitarias y las de aceptación. Las pruebas unitarias son definidas por los programadores antes de comenzar a escribir código. Éstas deben contemplar cada módulo del sistema que pueda generar fallas. Para poder integrar el código realizado al ya existente, el mismo debe aprobar satisfactoriamente todos los casos de prueba definidos. El cliente con ayuda del tester define las pruebas de aceptación para cada historia de usuario a principio de cada iteración. Las pruebas de aceptación se utilizan para validar que cada requerimiento implementado funciona como se había especificado.

* ***Prácticas del equipo:***

1. *Metáfora*: La metáfora le brinda al equipo una imagen del sistema la cual ellos pueden utilizar para describir la estructura en forma simple y sencilla. Las ventajas de su aplicación es que hace más fácil la comprensión del sistema y colabora para mantener un diseño simple.
2. *Programación por pares*: Todo el código producido en XP es realizado en parejas, dos personas frente a una computadora. Los roles en la pareja son de “conductor” y “explorador”. El “conductor” es el que maneja el teclado y el ratón (mouse), pensando la mejor manera de cómo implementar el código. El “acompañante” tiene como tarea observar y corregir los errores cometidos por el “conductor”, considerar soluciones alternativas y sugerir nuevos casos de prueba. Esta constante revisión produce código y un diseño con mayor calidad. Periódicamente se debe intercambiar los roles de la pareja. De esta forma, se asegura transmitir el conocimiento del sistema a todo el equipo y eliminar las dependencias de personas que conocen partes del sistema.
3. *Propiedad colectiva*: Esta práctica se basa en que todo el código desarrollado pertenece al equipo. Todos los integrantes del equipo tienen la misma responsabilidad sobre todo el sistema. Al ser el equipo el responsable, cualquier integrante está autorizado a realizar los cambios que se consideren necesarios para mejorar la calidad del mismo. De esta manera se logra descentralizar el conocimiento y no generar dependencias hacia las personas.
4. *Integración continua*: La integración de código se debe realizar en forma continua, esto implica al menos en un lapso de un día de trabajo y de ser posible cada una pocas horas. La ventaja de la integración continua es obtener retroalimentación lo más rápido posible. Además de ser más sencilla que las integraciones que se realizan luego de varias semanas de programación.
5. *40 horas semanales*: Esta práctica expone que 40 es la cantidad de horas que se debe dedicar al trabajo semanalmente. De esta manera se logra cometer menos errores por cansancio y aumentar la productividad del equipo.

* ***Prácticas del proceso:***

1. *Cliente on-site*: Para lograr un correcto funcionamiento de un proyecto en XP es necesario contar con la presencia full-time del cliente en el lugar de trabajo, siendo éste un integrante más del equipo. Contar con un cliente en el lugar de trabajo que fomenta la comunicación y reduce la posibilidad de malentendidos y tiempos de espera.
2. *Planificación*: Mediante esta práctica se realiza la planificación del proyecto en XP. La misma consiste en el plan de entregas, el plan de iteraciones, las historias de usuario, las tareas y los casos de pruebas para las mismas.
3. *Pequeñas entregas*: Las entregas deben ser lo más pequeñas posibles, conteniendo siempre los requerimientos del negocio más importantes para el cliente en ese momento dado. De esta manera el cliente va obteniendo funcionalidades del sistema en forma gradual hasta finalizar el proyecto. En cada entrega los programadores obtienen retroalimentación del cliente determinando si lo implementado es lo que en realidad necesita.

**Gerencia de la Calidad como marco para la medición de la Madurez de las Capacidades de los Procesos de Proyectos Desarrollo**

La gerencia de la calidad tiene como objetivo establecer un sistema gerencial que evite fallas en el ciclo de operación de la compañía, (Crosby, 1979) y es definido por la ISO (2005: 19) como “Actividades coordinadas para dirigir y controlar una organización en lo relativo a la calidad”.

En vista que uno de los objetivos de la gerencia de la calidad es “prevenir y reducir los problemas de calidad de una manera efectiva, se obtiene como beneficio la cuantificación de los costos derivados de la [misma]” (Cokins, 2006). La precisión de los “…costos de la calidad en la gerencia de las organizaciones de desarrollo, es uno de los elementos que determina el nivel de madurez de la gerencia de la calidad” (Crosby, 1979); por lo que su compresión y aplicación se constituye en prácticas sustantivas y operativas para la generación de valor y nuevas capacidades organizacionales.

Según Juran, Medina y Ballester (1990), la gerencia de la calidad se realiza por medio de tres procesos conocidos como la *Trilogía de Juran*: a) planificación de la calidad, que contempla las actividades de desarrollo de los productos y servicios requeridos por los clientes; b) control de la calidad, que evalúa el comportamiento real de la calidad, lo compara con los objetivos y actúa sobre las diferencias; y c) mejora de la calidad, que busca aumentar la calidad a niveles superiores a los actuales.

Como parte de las lecciones aprendidas de la gestión de la calidad aplicada en Japón, Deming (1982), definió los principios para transformación de la industria americana en catorce puntos: 1) Crear constancia en la mejora de productos y servicios; 2) Adoptar una nueva filosofía; 3) Desistir de la dependencia en la inspección en masa; 4) Terminar con la práctica para comprar con más bajo precio; 5) Mejorar constantemente; 6) Establecer entrenamiento dentro del trabajo; 7) Establecer líderes; 8) Eliminar el miedo y construir la confianza; 9) Eliminar las barreras entre los departamentos; 10) Eliminar eslóganes, exhortaciones y metas pidiendo cero defectos o nuevos niveles de productividad; 11) Eliminar cuotas numéricas y la gestión por objetivos; 12) Remover barreras para apreciar la mano de obra y los elementos que privan a la gente de la alegría en su trabajo; 13) Instituir un programa vigoroso de educación y auto mejora; y 14) poner a todos en la compañía a trabajar para llevar a cabo la transformación.

De igual manera, la ISO (2005), identifica ocho (8) principios de la gerencia de la calidad, con propósito de guiar a las organizaciones hacia un mejor desempeño: 1) Enfoque en el cliente; 2) Liderazgo; 3) Participación del personal; 4) Enfoque basado en procesos; 5) Enfoque de sistemas para la gestión; 6) Mejora continua; 7) Enfoque basado en hechos para la toma de decisiones; y 9) Relaciones mutuamente beneficiosas con el proveedor.

La evolución de la calidad llevo este concepto a la denominada Gerencia de la Calidad Total, que es referida cuando la calidad se extiende en toda la organización alcanzado un nivel estratégico y abarcando no solo productos, sino también los procesos como la gerencia del talento humano, forma de organización, estandarización de procesos y enfocándose en la satisfacción de los clientes (Juran, 1995). Cuando las organizaciones llegan al nivel de la calidad total ponen en prácticas, enfoques y modelos de mejoramiento continuo, círculos de la calidad, trabajo en equipo y comienzan a trabajar en el concepto de calidad desde el diseño de los procesos y productos (Cuatrecasas, 2005).

**Capacidad del Proceso de Desarrollo de Sistemas de Software**

Según la Real Academia Española, un proceso, en una definición muy general, “es la acción de avanzar o ir adelante, al paso del tiempo y al conjunto de etapas sucesivas”. (RAE, 2001). El desarrollo de los pasos de un proceso tiene un fin determinado, el producto que será entregado al cliente. Un proceso “es una secuencia de actividades mediante la cual se producen bienes (Productos y/o servicios)” (Chase, Jacobs y Aquilano, 2009).

Desde la perspectiva de la calidad, es relevante que cada paso o etapa sea efectiva, es decir, que no se generen retrabajos o desechos que hagan que el proceso genere productos defectuosos. Por tanto, un proceso que no genere defectos o inconformidades es un proceso con capacidad para desempeñarse conforme a las necesidades del cliente o de acuerdo con los requisitos de diseño.

De lo que antecede, se platea que la capacidad de proceso es concebida como el nivel de variación con respecto al cumplimiento del requisito establecido. Esta variabilidad está relacionada con las fluctuaciones naturales que existen en el proceso o variaciones inherentes a él. De acuerdo con esto, Juran señala que la capacidad del proceso es “la medición de la variación inherente del producto que es resultado de un proceso” (Gryna, Chua, y Defeo, 2007).

Montgomery (1991), define la capacidad de proceso como, “la forma en que se compara la variabilidad inherente de un proceso con las especificaciones o requerimientos del producto”. Es a partir de la relación de estos, cuando se comienzan a determinar los índices de capacidad del proceso.

El análisis de la capacidad de un proceso es concebida por Montgomery, (1991: 237), como al conjunto de técnicas estadísticas utilizadas para cuantificar la variabilidad del proceso, analizar esta variabilidad en relación con los requisitos o especificaciones del producto, y para ayudar en el desarrollo y la manufactura, eliminando o reduciendo en gran medida esta variabilidad. Es de vital importancia para el programa de control de calidad. Entre los principales usos del análisis de capacidad del proceso están los siguientes (Montgomery, 1991):

* Ayuda a modificar o rediseñar el proceso.
* Auxilia a especificar los requerimientos que debe de cumplir el equipo.
* Asiste a seleccionar el mejor proveedor.
* Apoya a predecir si el producto cumplirá con las especificaciones del cliente.
* Ayuda a planear la secuencia del proceso de producción cuando existe un efecto del proceso en las tolerancias.
* Asiste a reducir la variabilidad en un proceso de manufactura (237).

De acuerdo con Paulk (1994), la capacidad de un proceso es concebido como “el rango de resultados esperados que se pueden obtener tras seguir un [modelo de madurez]”. En términos aplicados a la medición de la capacidad de un proceso, permite a la gerencia medir la reproducibilidad de un proceso y se atribuye que la mayoría de las dificultades para el cumplimiento con las especificaciones o necesidades de un proceso, no están en los errores o causas especiales sino en el propio sistema de gestión, a lo que se denomina responsabilidad de la gerencia.

Juran, Gryma y Bingham (2005), relacionaron el concepto de la capacidad del proceso (nivel de variación y costos de la calidad) con la planificación, control de calidad y mejora continua a través de la denominada *Trilogía de Juran*. La cual se realiza, según Juran, Medina y Ballester (1990), por medio de tres procesos, tales como: a) *Planificación de la calidad*: que contempla las actividades de desarrollo de los productos y servicios requeridos por los clientes; b) *Control de la calidad*: que evalúa el comportamiento real de la calidad, lo compara con los objetivos y actúa sobre las diferencias; y c) *Mejora de la calidad*: que busca aumentar la calidad a niveles superiores de los actuales.

**Modelos de Madurez para la evaluación de las Capacidades de los procesos de Desarrollo de Sistemas de Software**

Un modelo de madurez de capacidades es considerado como una “…representación simplificada de los elementos esenciales de los procesos eficaces” (Paulk, al et., 1996), este concepto fue tocado por primera vez por Crosby (1979) en su representación del Cuadro de Madurez de la Gestión de la Calidad o *Quality Management Maturity Grid*, tiene como base los conceptos de control estadístico de procesos y mejora continua desarrollados por Shewhart (Paulk, al et., 1996).

Luego el concepto de un modelo de madurez de la capacidad de los procesos fue desarrollado por el *Software Engineering Institute* (SEI) en la Carnegie-Mellon University como requerimiento del Departamento de Defensa del Gobierno Federal de los Estados Unidos de América para evaluar los procesos de desarrollo de software en el año 1987, denominado Modelo de Madurez de Capacidades para el Desarrollo de Software o *Capability Maturity Model for Software* (SW-CMM). A partir de este modelo y en combinación con el ISO/IEC 15504 (ahora ISO/IEC 33000), nace en el año 2001 el Modelo de Madurez de Capacidades Integrado o *Capability maturity model integration* (CMMI), que es un modelo para la mejora y evaluación de procesos para el desarrollo, mantenimiento y operación de sistemas de software (SEI, 2010).

Este modelo es un marco de referencia que permite realizar una medición del proceso de desarrollo de software a través de cinco niveles, lo que permite a las organizaciones tener una guía de los pasos a seguir hacia mejora de sus procesos (Myung, 2009). El uso de los niveles permite determinar que procesos se están relegando en el camino hacia la madurez (Crosby, 1979), lo cual ha sido adoptado por muchas organizaciones como forma de realizar una autoevaluación para determinar las estrategias a seguir (Wiele, al et., 2000).

CMMI contempla un modelo de Madurez en niveles, áreas de procesos claves, características comunes y prácticas claves, con el propósito de ofrecer una guía detallada del camino a seguir y suministra una base concreta para la evaluación y autoevaluación de los procesos. Este punto de desarrollo ha permitido la evaluación positivamente tanto en aplicación como en aceptación de este modelo.

Para CMMI los niveles de Madurez son: incompleto, realizado, gestionado y definido (SEI, 2010). Los niveles de madurez indican los objetivos a alcanzar y a excepción del nivel 1 indica las áreas de procesos claves en las que la organización debe enfocarse en mejorar, en total para el CMMI se identificaron dieciocho procesos claves y a través de las características comunes se podían determinar si estos han sido implementados exitosamente, teniendo como base las prácticas claves que indican la infraestructura y actividades claves a realizar para alcanzar una eficiente y efectiva implementación, (Paulk, al et, 1996).

**Definición de términos básicos**

**Actividad**: Cualquier paso o función que se realiza (mental o físicamente) para alcanzar algún objetivo. Incluyendo todo el trabajo realizado para realizar las tareas del proyecto y la organización.

**Áreas Clave del Proceso**: Grupo de actividades relacionadas que cuando se llevan a cabo en conjunto alcanzan un conjunto de metas (consideradas importantes para aumentar la capacidad del proceso). Las ACP son al proceso de desarrollo de software lo que los cimientos a una casa. Cada una de las dieciocho ACP pertenece a uno y solo uno de los cinco niveles de madurez. Como resultado se obtiene un despliegue de los procesos de software que son efectivos, usables y aplicados constantemente a lo largo de la organización.

**Capacidad**: Habilidad que tiene algo o alguien para cumplir con las exigencias, objetivos o metas.

**Capacidad del proceso**: Es la aptitud del proceso para producir productos dentro de los límites de especificaciones de calidad.

**Capacidad medida**: Se refiere al hecho de que la capacidad del proceso se cuantifica a partir de datos que, a su vez, son el resultado de la medición del trabajo realizado por el proceso.

**Capacidad inherente**: Se refiere a la uniformidad del producto que resulta de un proceso que se encuentra en estado de control estadístico, es decir, en ausencia de causas especiales o atribuibles de variación.

**CMMI**: Siglas del modelo de madurez de capacidad integrado, no lleva las siglas SW (que representa la palabra software) por ser un modelo que abarca a toda la organización.

**Habilidad**: Esta palabra se usa en el sentido de aptitud, basada en el desempeño probado, para lograr resultados que se puedan medir.

**Madurez de un proceso de desarrollo de software**: Es el punto hasta el cual un determinado proceso es explícitamente definido, administrado, medido, controlado y efectivo.

**Madurez**: es el potencial de crecimiento de la capacidad del proceso que indica tanto la riqueza del proceso de desarrollo de software de una organización, como la consistencia con la cual es aplicado en los proyectos a lo largo de la misma.

**Nivel de madurez**: Plataforma bien definida desde la cual se puede obtener un proceso maduro de software.

**Proceso**: Se refiere a alguna combinación única de máquinas, herramientas, métodos, materiales y personas involucradas en la producción.

**CAPÍTULO III**

**MARCO METODOLÓGICO**

**Naturaleza de la Investigación**

La comunidad científica a través del tiempo desarrolló el conocimiento formal y originó los enfoques de investigación clásicos. No obstante, el avance tecnológico demanda nuevos procedimientos o la innovación de los existentes para el abordaje de las situaciones problemáticas para que se facilite la solución a esas necesidades.

En ese contexto, Cross, Naughton y Walker (1981), consideraron al método científico como un modelo de solución que indaga la naturaleza de los problemas y definieron un método innovador basado en el diseño, dirigidos al desarrollo y la innovación. Hevner y Chatterjee (2010), señalan que las actividades asociadas al diseño tienen una larga trayectoria pero fueron introducidas en la computación y tecnologías de la información en la década de 1990, en el *Workshop on Information Technology and Systems (WITS) and Electronic Group Decision Support Systems* *(GDSS) of the University of Arizona*, se conoció su relevancia para la investigación en el ámbito de las ciencias de la computación.

Por su parte González (2007: 3), asevera que la Ciencia del Diseño “designa conocimientos específicos que son elaborados para resolver, de manera articulada, problemas concretos que surgen en el entorno humano”, además “identifica un nuevo campo del saber…”, abarca un conjunto de prácticas científicas propias.

El presente estudio se ubica dentro del enfoque de ciencias del diseño, que Simon (1996: 111), define como un paradigma de investigación pragmático que aboga por la creación de artefactos innovadores para la resolución de problemas del mundo real, tiene sus raíces en la ingeniería y las ciencias de lo artificial, es decir, se inclina a la resolución práctica de problemas.

Por su parte Vaishnavi y Kuechler (2004), señalan que la investigación bajo este enfoque consiste en la creación de nuevos conocimientos a través del diseño de artefactos (modelos, procesos, entre otros) novedosos o innovadores y del análisis de su uso y desempeño.

Este enfoque se focaliza en la necesidad de artefactos que no existen para satisfacer una necesidad humana. Por lo que, los enfoques tradicionales son considerados inadecuados, debido que están orientados al estudio de cosas o fenómenos que existen, por esta razón se considera ideal este paradigma para el presente estudio, es decir, para el diseño de un modelo integrado para la evaluación de la madurez de las capacidades de los métodos ágiles de los proyectos de desarrollo de software.

**Diseño de la Investigación**

En correspondencia al diseño de la investigación, Balestrini (2002), señala que “es la estrategia o plan que guía la investigación”. Para Espinoza (2012) señala que debido a las diferencias que “radican en la naturaleza de las ingenierías, incluida la Ingeniería del Software, respecto de otras disciplinas empíricas y formales, dificultan la aplicación directa de los métodos de investigación tradicionales”. Por consiguiente, han surgido nuevas propuestas metodológicas como el Modelo de Proceso general para la investigación en la Ciencia del Diseño propuesto por Vaishnavi y Kuechler (2004); Vaishnavi y Kuechler (2015: 12).

**Procedimiento de la Investigación**

El Modelo de Proceso general para la investigación en la Ciencia del Diseño propuesto por Vaishnavi y Kuechler (2004); Vaishnavi y Kuechler (2015: 12), está compuesto por cinco etapas (Vaishnavi y Kuechler, 2004), tales como (ver figura 4):

**Fase I. Conocimiento del problema (Awareness of Problem)**: esta la fase inicial del proceso de investigación, consiste en el descubrimiento de un problema o necesidad como el planteado en el presente estudio.

**Fase II. Propuesta (Suggestion)**: constituye la elaboración de un modelo, diseño tentativo o prototipo de solución al problema planteado, esta solución puede ser un planteamiento completamente nuevo o una nueva funcionalidad para una propuesta existente. Para el presente estudio, la propuesta de solución está comprendida por el alcance de los objetivos de la investigación, donde el objetivo principal es el diseño de un modelo integrado para la evaluación de la madurez de las capacidades de los métodos ágiles de los proyectos de desarrollo de software.

**Fase III. Desarrollo (Development)**: se construye o desarrolla la propuesta, las técnicas para su implementación variaran dependiendo del artefacto que se desea crear, el artefacto puede ser un modelo, algoritmo, método, sistema, entre otros; la novedad del proceso está en la elaboración de la propuesta, la ejecución de la misma puede ser común, dependiendo del estado de la práctica del artefacto.

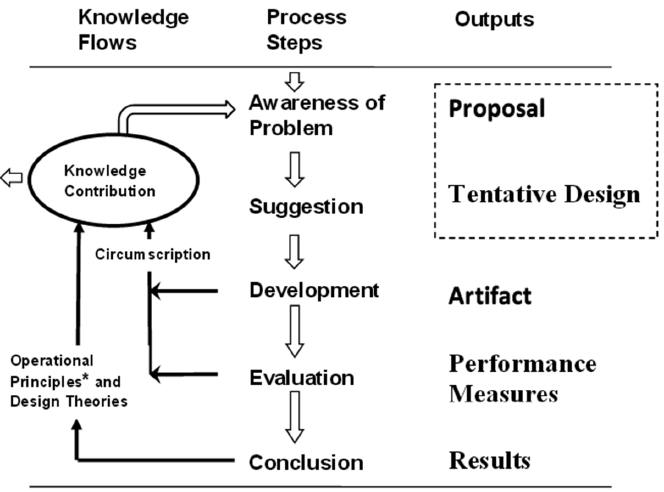


Figura 4. **Modelo general de un ciclo de investigación en ciencia del diseño**.

Fuente: adaptación de Vaishnavi y Kuechler (2004).

Para efectos del diseño un modelo integrado para la evaluación de la madurez de las capacidades de los métodos ágiles de los proyectos de desarrollo de software, se utiliza como guía la *Ingeniería del Método Situacional* (ver figura 5).

Este modelo es utilizado en múltiples estudios de investigación, además es interesante por el enfoque orientado a las ciencias de la computación, por lo tanto, fue seleccionado como procedimiento para desarrollar el presente estudio.

De acuerdo a Harmsen, Brinkkemper y Oei (1994: 6), la *Ingeniería de Método Situacional* es un enfoque ingenieril, conducente a la definición de métodos para el diseño de sistemas en una situación específica, por lo que se considera adecuado para el presente estudio.

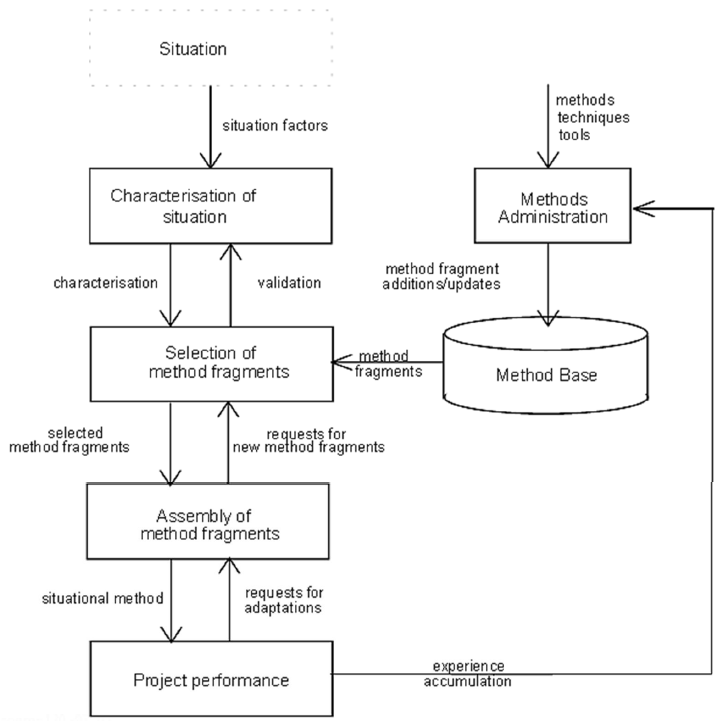


Figura 5. **Proceso genérico de la Ingeniería de Método Situacional.**

**Fuente**: Harmsen (1997)

Las actividades a realizar, según el enfoque basado en extensión para el diseño de un modelo integrado para la evaluación de la madurez de las capacidades de los métodos ágiles de los proyectos de desarrollo de software, son las siguientes:

***1. Caracterizar la situación problemática***: en esta actividad se describe la situación actual del problema objeto en estudio, se analizan las tendencias presentes en cuanto a métodos, procesos y/o modelos reconocidos, probados y empleados en marco de la Ingeniería del Software. Para el caso específico del diseño de un modelo integrado para la evaluación de la madurez de las capacidades de los métodos ágiles de los proyectos de desarrollo de sistemas de software, se realizan las siguientes actividades:

1. Indagación si las organizaciones de desarrollo consideran algún método ágil de desarrollo como: AUP, Scrum o XP.
2. Precisión de las prácticas utilizadas por estas organizaciones en el desarrollo de proyectos ágiles de sistemas de software.

***2. Selección de fragmentos de métodos:***

Dentro de este enfoque, se utiliza en particular el *Enfoque Basado en Extensión* (ver tabla 1), que según Henderson-Sellers, Sellers y Ralyté (2010), permite definir un nuevo método completando los vacíos de un método existente con elementos de otros métodos y omitiendo los elementos irrelevantes.

Tabla 1. **Enfoques para la creación de Métodos en la Ingeniería de Métodos Situacional.**

|  |  |
| --- | --- |
| Enfoque | Descripción |
| Basado en Ensamblaje | Un método nuevo se construye a partir de fragmentos de métodos ya existentes. Los fragmentos pueden ser categorizados en fragmentos de producto y fragmento de procesos. Este enfoque sigue la estrategia de reutilización. |
| Basado en la Extensión | El método requerido es creado a partir de un método ya existente el cual se extiende haciendo uso de patrones y rellenando las faltas del método original. |
| Basado en Paradigma | Este enfoque se fundamenta en el concepto que un nuevo método se obtiene a partir de una abstracción de un método existente. Se le llamatambién enfoque basado en la evolución. |

**Fuente**: Espinoza (2012) con adaptación de Henderson-Sellers y Ralyte (2010)

Gehlert y Pfeiffer (2007), señalan que el enfoque basado en la extensión, se centra en un método existente y proporciona nuevos elementos al mismo. El objetivo de este enfoque es mejorar un método con las nuevas características que son útiles para satisfacer las necesidades del proyecto (ver figura 6).

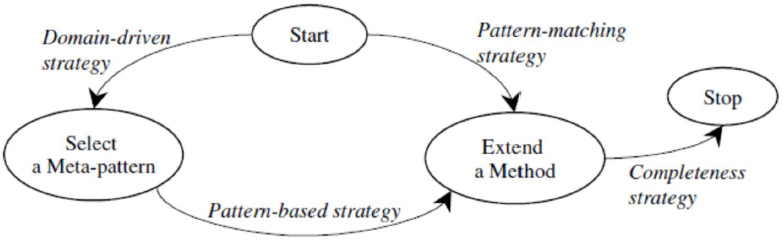


Figura 6. **Enfoque basado en Extensión de la ingeniería de método situacional.** Fuente: Ralyté, Deneckère y Rolland (2003)

El enfoque de extensión de la ingeniería de método situacional, al igual que los otros enfoques, contempla como etapa inicial la caracterización de la situación donde se desarrolla el proyecto, posteriormente según Henderson-Sellers y Ralyte (2010: 461), se deben realizar las siguientes actividades:

***2.1. Seleccionar el método base***: luego de la precisión de los métodos ágiles y de la determinación las prácticas usados por las organizaciones de desarrollo. Ahora se selecciona el método base que será extendido para el proceso de evaluación de la madurez y las capacidades propuesto. En ese sentido, se realizan las actividades mencionadas a continuación:

1. Análisis de los principios del Manifiesto Ágil en marco de los métodos ágiles de desarrollo en las realidades de la Industria del Software, que se encuentran disponibles en la literatura con el objeto de indagar cuáles principios aplican a los métodos ágiles de desarrollo de sistemas de software.
2. Comparación de los diferentes modelos de madurez para la evaluación de la madurez de las capacidades del proceso de desarrollo de sistemas de software que se encuentran disponibles en la literatura, para la selección de una opción entre varias evaluadas.

***2.2. Integrar los elementos de otros métodos para llenar las lagunas en el método base***:

1. Identificación de las áreas de procesos claves y los indicadores para la evaluación de la madurez las capacidades de los métodos ágiles de desarrollo de sistemas de software.
2. Definición de las directrices orientadas al diseño de un modelo integrado para la evaluación de la madurez de las capacidades conforme a los métodos ágiles de los proyectos de desarrollo de sistemas de software.

***2.3. Desarrollar nuevos trozos de métodos si no es posible integrarlos de métodos existentes:*** Una vez definidas las directrices orientadas al diseño de un modelo integrado para la evaluación de la madurez de las capacidades conforme a los métodos ágiles de los proyectos de desarrollo de sistemas de software, se adaptan los elementos de los métodos ágiles de los proyectos de desarrollo de sistemas de software al modelo integrado para la evaluación de la madurez de las capacidades.

***3. Ensamblaje de los fragmentos de métodos, lo que genera un método situacional.*** A partir de correlación de elementos, se procede al diseño del modelo integrado para la evaluación de la madurez de las capacidades de los métodos ágiles de los proyectos de desarrollo de sistemas de software.

***4. Desempeño del proyecto, que se evidencia con la experiencia de uso:*** no aplica alpresente estudio.

**Fase 4. Evaluación (Evaluación)**: una vez diseñado el modelo propuesto, se define el proceso de evaluación cuali-cuantitativa de la madurez y la capacidad de los métodos ágiles de los proyectos de desarrollo de sistemas de software.

**Fase 5: Conclusiones**: esta fase finaliza el presente estudio, el cual genera conocimientos a la comunidad científica. En este contexto, se plantea que el modelo propuesto sea una solución aproximada a la realidad, a partir de la cual podrían surgir nuevas investigaciones en torno y respecto al problema objeto de estudio. Las conclusiones de la presente investigación se presentan en Capítulo V de este trabajo.

**Referencias Bibliográficas**

Ambler S. (2006). The Agile Unified Process (AUP). [Online]. Available in: http://www.ambysoft.com/unifiedprocess/agileUP.html. [Consulta: 15/09/2018].

Ali, J., Chew, T. G., & Tang, T. C. (2004). Knowledge management in agile organizations. Sunway Academic Journal, 1, 13–20.

Alonso J. (2017). Modelo CMMI y métodos agiles en la gestión de proyectos de software. Maestría interuniversitario en Dirección de Proyectos, Departamento de Explotación y Prospección de Minas, Universidad de Oviedo, Oviedo, España.

Bauer, F. (1968). Report of a conference sponsored by the NATO Science Committee. NATO Software Engineering Conference. Garmisch, Germany: P. Naur and B. Randell.

Balestrini, M., (2002). Cómo se elabora el Proyecto de Investigación. Tercera Edición. Caracas: B.L Consultores.

Beck, K. (1999). Embracing change with extreme programming. Computer, 32(10), 70-77.

Beck, K., Beedle, M., Bennekum, A. v., Cockburn, A., Cunningham, W., Fowler, M., y otros. (2001). Manifesto for Agile Software Development. [Online]. Disponible en: http://agilemanifesto.org/ (Consultado: 23/09/2018).

Boiero M. y Estrada G. (2008) Interdisciplinariedad y Multidisciplinariedad: Un Ensayo en la enseñanza de Finanzas y Administración Pública [Online]. Disponible: www.angrad.org.br/...multidisciplinariedad.../download/. [Consulta: 30/08/2018].

Carmona M. (2004). Transdisciplinariedad: Una propuesta para la Educación Superior en Venezuela. [Revista-Online]. Revista de Pedagogía, V. 25, Nro. 73, Caracas, Venezuela. Disponible en: http://www.scielo.org.ve/scielo.php?pid=S0798-97922004000200007&script=sci\_arttext. [Consulta: 30/08/2018].

Cokins, G. (2006). Measuring the Cost of Quality for Management. Quality Progress, Vol (39), 45-51.

Cross, N., Naughton, J. y Walker, D. (1981, Octubre) "Design method and scientific method". Design Studies. Vol. 2. No. 4. pp. 195-201.

Crosby, P. (1979). Quality is Free, The Art of Making Quality Certain. New York, NY: Mc Graw-Hill.

Cuatrecasas, L. (2005). Gestión Integral de la Calidad: Implantación, Control y Certificación.Barcelona, España: Edición 2000.

Deming, E. (1982). Out of the Crisis. Massachusetts, USA: MIT Press.

Espinoza, G. (2012), en su trabajo de grado de maestría “Método de validación de requisitos funcionales de software a partir de prototipos de interfaces de usuario basados en patrones RIA. Maestría en Ciencias de la Computación, Decanato de Ciencias y Tecnología, UCLA, Barquisimeto, Venezuela.

Firestone, J. (2001). Knowledge and innovation. [Online]. Journal of the Knowledge management consortium international, inc. Volume one, no. Two, january 15. Disponible en: www.kmci.org/media/FirestoneKIv1n2.pdf. [Consulta: 3/09/2018].

Gryna F., Chua R. & DeFeo J. (2007). Método Juran. Análisis y planeación de la calidad. México: McGraw-Hill Interamericana.

Guzmán J.C. (2010). Modelo de gerencia de proyectos para la Dirección de Automatización, Informática y Telecomunicaciones de Pdvsa, desde la gerencia del conocimiento. Trabajo de Grado de Maestría en Gerencia de las TIC, UNEFA, Caracas, Venezuela.

Harmsen, A., Brinkkemper, S. y Oei, H. (1994). Situational Method Engineering for Information System Project Approaches. In: A.A. Verrijn Stuartand T.W. Olle (Eds.), Methods and Associated Tools for the Information Systems Life Cycle. Netherlands, September 1994, (pp. 169-194).

Hass, K. B. (2007).The blending of traditional and agile project management. PM World Today, 9(5). Retrieved August 30, 2009, Available in: http://www.pmforum.org/library/tips/2007/PDFs/Hass-5-07.pdf. [Consulta: 3/09/2018].

Henderson-Sellers, B. y Ralyté, J. (2010) Situational Method Engineering: Stateof-the-Art Review. Journal of Universal Computer Science, vol. 16, no. 3 (2010), (pp.424-478) [En línea] Disponible: http://goo.gl/KnJqOz [Consulta: 014/09/2018].

Hevner, A., and Chatterjee, S. (2010) Design Research in Information Systems. Theory and Practice. Springer. New York. pp. 9-21.

IEEE. (1999). IEEE Std. 610-1990: Standard Glossary of Software Engineering Terminology, IEEE Standards Software Engineering, Volume 1.

Iglesias, A., Messino, A., Pedroza, P., & Llanos, R. (2015). Modelo aplicable a la gestión de procesos ágiles de desarrollo de software basado en CMMI–DEV 1.3 y scrum. Investigación e Innovación en Ingenierías, 3(1).

International Organization for Standardization [ISO]. (2003). ISO/IEC 15504-2, Information technology— Process assessment — Part 2: Performing an assessment. International Standards Organization.

International Organization for Standardization [ISO]. (2005). Norma Internacional ISO 9000 - Sistemas de gestión de la calidad – Fundamentos y Vocabulario. Ginebra, Suiza: ISO/TC 176/SC 1.

International Organization for Standardization [ISO]. (2008). Norma Internacional ISO 9001 - Sistemas de gestión de la calidad – Requisitos. Ginebra, Suiza: ISO/TC 176/SC 2.

International Organization for Standardization [ISO]. (2011). ISO/IEC 25010: Software engineering-Software product Quality Requirements and Evaluation (SQuaRE), Quality model.

Iribarren O. (2012). Modelo de Mejora Continua para la Gestión de Requisitos de Software usando Métodos Ágiles. Maestría en Ciencias de la Computación, Decanato de Ciencias y Tecnología, UCLA, Barquisimeto, Venezuela.

Jacobs R., Chase R. and Aquilano N. (2009). Operations and Supply Management. US: McGraw Hill

Jacobson I., Booch G. & Rumbaugh J. (1999).The Unified Software Development Process, Massachuset, USA, Editorial: Addison-Wesley.

Juran, J. M., Medina, J. N., & Ballester, M. G. (1990). Juran y el liderazgo para la calidad: manual para ejecutivos. Ediciones Díaz de Santos.

Juran, J. (Ed) (1995). A History of Managing for Quality: The Evolution, Trends, and Future Directions of Managing for Quality. Milwaukee, WI: ASQ Quality Press.

Juran J, Gryma F. & Bingham R. (2005). Manual de Control de la Calidad 3ra Ed. Barcelona, España: Editorial Raverté.

Martínez M. (2003). Transdisciplinariedad y Lógica Dialéctica: Un Enfoque para la Complejidad del Mundo Actual. [Online]. Proc. Conciencia Activa 21. 2003, 1, pp. 107-146. Disponible en: http://prof.usb.ve/miguelm/transdiscylogicadialectica.html. [Consulta: 30/08/2018].

Martínez M. (2007). Transdisciplinariedad e Investigación en la Educación Superior. Proc. de I Congreso Internacional de Calidad e Innovación en Educación Superior. Caracas, Venezuela, 9-13 de abril de 2007.

Montgomery, D. (1991). Control Estadístico de la Calidad. Mexico D.F., Mexico: Limusa.

Morin E. (1998), Articular los saberes. ¿Qué saberes enseñar en las escuelas?, Buenos Aires, Argentina, Editorial:Universidad del Salvador.

Myung S. (2009). Process Quality Levels of ISO/IEC 15504, CMMI and K-model. International Journal of Software Engineering and Its Applications, Vol (3), 33-42.

Nicolescu, B. (1997) The Transdisciplinary Evolution of the University Condition for Sustainable Development. [Online]. Talk at the International Congress Universities Responsabilities to Society, Bangkok, Thailand, 1997, November 12-14. Disponible en: http://perso.club-internet.fr/nicol/ciret/bulletin/b12/b12c8.htm. [Consulta: 30/08/2018].

Larman, C. (2002). Applying UML and Patterns: An Introduction to OO analysis and design and the Unified Process, Second Edition, Boston, USA, Editorial: Prentice Hall.

Larman, C. (2004). Agile and iterative development: A manager’s guide. Boston, MA: Pearson Education.

Object Management Group [OMG]. (2008). Software & Systems Process Engineering Meta-Model Specification (SPEM). Version 2.0, Document Number: formal/08-04-02, Massachusetts, U.S., Editorial: OMG. Disponible en: http://www.omg.org/spec/SPEM/2.0/PDF. [Consulta: 27/09/2018].

Owen, R., Koskela, L.J., Henrich, G., & Codinhoto, R. (2006). Is agile project management applicable to construction? In R. Sacks& S. Bertelsen (Eds.), Proceedings of the 14th Annual Conference of the International Group for Lean Construction (pp. 51–66).Santiago, Chile: Ponteficia Universidad Católica de Chile.

Paulk M. (1994). A comparison of ISO 9001 and the Capability Maturity Model for Software, Technical Report CMU/SEI-94-TR-012.

Paulk M., Curtis B., Chrissis M. & Weber C. (1996). Capability Maturity Model for Software, Version 1.1 (CMU/SEI-93-TR-024). Pittsburgh, USA: Carnegie Mellon University.

Project Management Institute [PMI] (2017). A Guide to the Project Management Body of Knowledge (PMBOK Guide). Sixth Edition, Pennsylvania, USA: Project Management Institute Inc.

Real Academia Española [R.A.E.] (2001). Diccionario de la lengua española [Dictionary of the Spanish Language] (22nd ed.). Madrid, Spain: Author.

Rodríguez P. (2008). Estudio de la aplicación de metodologías ágiles para la evolución de productos de software. Tesis de Máster en tecnologías de la información, Universidad Politécnica de Madrid.

Rodríguez L. (2014). Proceso de Desarrollo de Software basado en metodologías ágiles y CMMI-DEV. Maestría Universitaria en Investigación en Ingeniería de Software y Sistemas Informáticos. Departamento de Ingeniería de Software y Sistemas Informáticos, Universidad Nacional de Educación a Distancia, universidad pública de España. Madrid-España.

Rodríguez H. (2017). Análisis y diseño de un modelo con integración de una metodología ágil en el nivel 2 de CMMI. Maestría en Ingeniería – Telecomunicaciones, Facultad de Ingeniería, Universidad Nacional de Colombia, Bogotá, Colombia.

Simon, H. (1996) The sciences of the artificial. 3rd edition. The MIT Press, Cambridge, MA. (p. 11). [Libro en línea] Disponible: http://m.friendfeedmedia.com/092e5a73c91e0838eeb11e0fe90edaf9e9afc065 (Consulta: 16/09/2018)

Sommerville, I. 2005. Ingeniería de Software. Pearson Educación. España.

Software Engineering Institute [SEI]. (2010). CMMI para el Desarrollo Versión 1.3 (CMU/SEI-2010-TR-033).Pennsylvania, PA: Carnegie Mellon University.

Standish Group International [SGI]. (2014). CHAOS MANIFESTO 2013: Think Big, Act Small. [Online]. Disponible en: https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/GENREF/S130301C.pdf. [Consultado: 22/09/2018]

Sutherland, J. V., & Schwaber, K. (1995). The SCRUM methodology. In Business object design and implementation: OOPSLA workshop.

Takeuchi, H., & Nonaka, I. (1986). The new new product development game. Harvard business review, 64(1), 137-146.

Tamayo y Tamayo M. (2004). El Proceso de Investigación Científica. (3era ed.). México. Editorial: Limusa S.A.

Teje M. y Buenen M. (2013). World Quality Report 2013-2014. [Online]. Disponible en: https://es.slideshare.net/capgemini/world-quality-report-2013-14 (Consultado: 22/09/2018).

Vaishnavi, V. y Kuechler, W. (2004). “Design Science Research in Information Systems,” (Online). Disponible: http://www.desrist.org/design-research-in-information-systems/ [Consulta: 09/09/2018].

Vaishnavi, V. and Kuechler, W. (2015). “Design Science Research Methods and Patterns: Innovating Information and Communication” USA: Taylor & Francis Group. [En línea] Disponible: http://books.google.co.ve/books?id=sI2Y9Jh8tq8C&printsec=frontcover#v=onepage&q&f=false. [Consulta: 15/09/2018].

Wiele T., Brown A., Millen R. & Whelan D. (2000). Improvement in Organizational Performance and Self-Assessment Practices by Selected American Firms, Quality Management Journal, Vol (7), 8-22.