



Universidad Popular Autónoma del Estado de Puebla
Centro Interdisciplinario de Posgrados
Investigación y Consultoría
Departamento de Ingeniería y Tecnologías de la
Información
Doctorado en Ingeniería de Software

**Modelo de mejora del proceso de capacitación para
desarrolladores de software en PYMES basado en
ambientes ágiles.**

Tesis que para obtener el Grado de Doctor
en Ingeniería de Software
Presenta

M.C. Carlos Enríquez Ramírez

Asesora: Dra. María del Pilar Gómez Gil
Investigadora Titular, INAOE

Puebla, México.

2013



UPAEP – Secretaría General

Dirección General de Apoyos Académicos

Dirección del Centro de Recursos para el Aprendizaje y la Investigación.

Biblioteca Central - **Karol Wojtyla**

Tesis Digitales Restricciones de uso:

DERECHOS RESERVADOS ©

PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis está protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de textos, imágenes, gráficas, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente de donde la obtuvo mencionando el autor o autores involucrados en el documento.

Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

ÍNDICE GENERAL

RESUMEN.....	
ABSTRACT.....	

Índice

Capítulo 1 INTRODUCCIÓN	10
1.1 Planteamiento del problema	12
1.2 Propósito de la investigación	13
1.3 Objetivo general	13
1.4 Objetivos específicos	13
1.5 Aportación	14
1.6 Justificación de la investigación.....	14
1.7 Alcances y limitaciones	14
1.8 Descripción del documento	15
Capítulo 2 MARCO TEORICO.....	15
2.1 Modelos de mejora de procesos	15
2.1.1 Modelos de mejora de procesos de software.....	17
2.1.2 Modelo CMMI.....	19
2.1.3 Patrón del área de procesos de CMMI.....	20
2.2 Capacitación en la ingeniería de software	21
2.2.1 Un modelo sistemático de capacitación.....	24
2.3 Métodos ágiles	27
2.3.1 Desarrollo dirigido por pruebas	29
2.3.2 Scrum	30
2.4 MoproSoft.....	34
2.4.1 Estructura de procesos de Moprosoft	35
2.4.2 Categorías de Moprosoft	36
2.4.3 Procesos de Moprosoft.....	36
2.5 Estado del arte.....	38
2.5.1 Algunos modelos de capacitación de desarrollo de software	40
Capítulo 3 METODOLOGÍA	42
3.1 Alcance y limitaciones.....	42

3.2	Hipótesis.....	43
3.3	Diseño de la investigación.....	44
3.4	Selección de la muestra.....	44
3.5	Recolección de datos	45
3.6	Aplicación del instrumento	48
Capítulo 4	EL MODELO PROPUESTO.....	49
4.1	El Modelo CAAM.....	49
4.2	Competencias a desarrollar.....	50
4.3	Estrategia de conformación del CAAM	51
4.4	Relación de actividades entre <i>Scrum</i> y <i>Moprosoft</i>	51
4.5	Relación de roles entre <i>Scrum</i> y <i>Moprosoft</i>	55
4.6	Niveles en el modelo de CAAM.....	56
4.7	Área clave de proceso para CAAM	57
4.7.1	Prácticas específicas de CAAM	57
4.8	Elementos de madurez en CAAM	61
4.9	Proceso de implementación de CAAM.....	63
Capítulo 5	RESULTADOS DE LA INVESTIGACIÓN.....	66
5.1	Resultados: análisis de datos	66
5.2	Resultados pre-capacitación.....	66
5.2.1	Análisis de confiabilidad de las encuestas de pre-capacitación.	66
5.2.3	Planeación de actividades para ambientes ágiles pre-capacitación.....	67
5.2.4	Comunicación con el cliente pre-capacitación.....	68
5.2.5	Uso de herramientas ágiles pre-capacitación.	68
5.2.6	Uso de herramientas de pruebas pre-capacitación.	69
5.2.7	Resumen de frecuencias de pre-capacitación.	69
5.3	Resultados post-capacitación.	69
5.3.1	Descripción de las características de post-capacitación.	70
5.3.2	Planeación de actividades para ambientes ágiles post-capacitación.	70
5.3.3	Comunicación con el cliente post-capacitación.	70
5.3.4	Uso de herramientas ágiles post-capacitación.	71
5.3.5	Uso de herramienta de pruebas post-capacitación.	71
5.3.6	Resumen de frecuencias de post-capacitación.....	71
5.4	Análisis de resultados de los cuestionarios pre- y post-capacitación.....	71

Capítulo 6 CONCLUSIONES	72
6.1 Trabajo futuro	74
Bibliografía	75
Anexo A. Cuestionario inicial para empresa	81
Anexo B. Comparación de Roles <i>Scrum</i> y Moprosoft en su proceso de “Administración de Proyectos Específicos”	86
Anexo C. Comparación de Roles Moprosoft y <i>Scrum</i> del proceso de “Desarrollo y Mantenimiento de Software”	90
Anexo D: Descripción Detallada de Competencias.....	94
Anexo E: Encuesta Inicial de implementación de CAAM.	96
Anexo F: Encuesta Final de implementación de CAAM.	99
Anexo G: Curso de validación de hipótesis del proceso de CAAM.....	103

ÍNDICE DE FIGURAS

Fig. 2-1 Etapas de adopción de procesos (Humphrey, 1989).....	16
Fig. 2-2 Niveles de madurez de procesos (Humphrey, 1989).....	18
Fig. 2-3 Fases de un diseño de capacitación (Buckley & Caple, 2009).	26
Fig. 2-4 Proceso de implementación de Scrum incluyendo TDD. Modificado de (Spada, 2005).	33
Fig. 2-5 Diagrama de categorías de procesos. (Oktaba, Alquicira, Su, & Martínez, 2005).....	38
Fig. 3-1 Grado de adopción de las metodologías obtenido en 38 empresas encuestadas.	46
Fig. 3-2 Nivel de adopción de metodologías ágiles obtenido en 38 empresas encuestadas.	46
Fig. 3-3 Motivos de adopción de las metodologías ágiles.....	47
Fig. 4-1 CAAM y las filosofías de trabajo Moprosoft, <i>Scrum</i> y desarrollo TDD.50	50
Fig. 4-2 División de procesos de Scrum a Moprosoft.....	53
Fig. 4-3 CAMM: Modelo de madurez para la capacitación.....	56
Fig. 4-4 Área clave de proceso de CAAM.	58
Fig. 4-5 Flujo de CAAM	61

ÍNDICE DE TABLAS

Tabla 4-1 Descripción de niveles para la funcionalidad de CAAM.	62
Tabla 4-2 Indicador de rendimiento de CAAM.	62
Tabla 5-1 Variables que integran la Planeación de actividades pre-capacitación.	68
Tabla 5-2 Variables que integran la Comunicación con el cliente pre-capacitación.	68
Tabla 5-3 Variables que integran el uso de herramientas ágiles pre-capacitación.	69
Tabla 5-4 Variable que muestra el Uso de herramientas de pruebas pre-capacitación.	69
Tabla 5-5 Variables que integran la Planeación de Actividades post-capacitación.	70
Tabla 5-6 Variables que integran la Comunicación con el cliente post-capacitación.	70
Tabla 5-7 Uso de herramientas ágiles de post-capacitación.....	71
Tabla 5-8 Diferencia porcentual de competencias post versus pre capacitación.	72

RESUMEN

Los procesos de formación de personal son vitales para las organizaciones, de las cuales dependen de la actualización y adquisición de habilidades para el conjunto eficiente de las actividades de desarrollo de software. En esta tesis se propone un nuevo modelo para el control de la madurez del proceso de formación de las empresas certificadas en la norma NMX-1-059/02-NYCE-2005 (conocido como Moprosoft) en el nivel 1, utilizando metodologías ágiles como Scrum. El modelo CAAM (Formación en entornos ágiles para Moprosoft) se basa en los niveles de madurez, la obtención de las habilidades a desarrollar en cada uno de los individuos de la organización, estos son la planificación de actividades para el medio ambiente ágil, la comunicación con el cliente, el uso de herramientas ágiles y el uso de herramientas de prueba .

ABSTRACT

The personnel training processes are vital for organizations, from them depend on the update and acquiring skills for the efficient set of the software development activities. In this dissertation we propose a new model to control the maturity of training process in certified companies in the standard NMX-1-059/02-NYCE-2005 (know how Moprosoft) at level 1 using agile methodologies, such as Scrum. The model CAAM (Training in Agile Environments to Moprosoft) it based in maturity levels, obtaining skills to be developed in each individuals in the organization, these are planning activities for agile environment, communication with customer, use of agile tools and use of testing tools.

Capítulo 1 INTRODUCCIÓN

La aplicación correcta de técnicas de ingeniería es una tarea difícil para las pequeñas y medianas empresas (PyMEs) dedicadas a la producción de software (Gresse von Wangenheim, Punter, & Anacleto, 2003). Aun así, una cantidad considerable del software que se produce en todo el mundo es construido por PyMEs, quienes a menudo operan con recursos limitados tanto económicos como de tiempo. Una gran mayoría de los productores de software, que aún no han implementado una metodología para la mejora de procesos de software, están pagando altos costos de producción y mantenimiento de sistemas, y por lo tanto están siendo desplazados en el mercado internacional, ya que no se encuentran en un nivel de competitividad igual a las empresas que poseen un método de desarrollo y una mejora de procesos (Herrera & Trejo, 2003).

La resistencia al cambio de empleados y/o administraciones de las empresas, así como las implicaciones generadas por la necesidad constante de ejecutar horas extras de trabajo, son factores que afectan negativamente el uso de metodologías. Para conseguir el cambio es necesaria una concientización de los beneficios del uso de procesos así como estar consciente de que el logro de un proyecto de software depende en gran medida de tener a la gente capacitada en las principales áreas claves de los procesos. A fin de formar una cultura donde el objetivo sea mejorar continuamente los procesos de desarrollo, es indispensable establecer mecanismos que permitan incorporar rápidamente a los profesionistas en las nuevas metodologías para asegurar que éstos las adopten. Por otro lado, las empresas se ven en la necesidad de gestionar con rapidez los cambios, acelerar el tiempo de comercialización, incrementar la productividad y mejorar la calidad con fin de enfrentar la competitividad, (VisionOne, 2010).

Algunas empresas desarrolladoras de software se han ocupado en los últimos años, de incorporar mejores prácticas de trabajo para el aseguramiento de su productividad. Esto también les ha permitido dar cumplimiento a la calidad demandada mediante la adopción de actividades, prácticas, roles específicos y normas de comportamiento. Los métodos ágiles son mecanismos de desarrollo de software

que cumplen con las especificaciones de rapidez de producción y calidad en el producto. Éstos aplican un desarrollo iterativo e incremental, promoviendo las entregas de manera evolutiva e incluyendo valores y prácticas adicionales a las tradicionales. Los autores y seguidores del “manifiesto ágil” aseguran que, siguiendo sus principios, se logra reducir los riesgos y mantener la precisión (Beck, Beedle, Bennekum van, & Cockburn, 2001). En los métodos ágiles, la planeación ocurre en cada iteración y se va alineando con las metas del negocio, además de que la inclusión del cliente en el desarrollo asegura que todos los requerimientos son levantados. Utilizando adecuadamente la filosofía “ágil” se pueden alcanzar metas de mejoras en la rapidez de entrega de proyectos, eficiencia y eficacia en el producto, sin afectar la calidad. La metodología ágil abarca aspectos humanos, organizacionales y tecnológicos del proceso de desarrollo del software (Dubinsky & Hazzan, 2003).

Uno de estos mecanismos de mejora es el entrenamiento sistematizado, que asegura el aprendizaje y la aplicación constante de nuevas tendencias de desarrollo. El profesionalismo es un ingrediente requerido en la cultura de una organización cuyo objetivo es la mejora de sus procesos (Radziwill N., 2005). Sin embargo, las PyMES desarrolladoras de software frecuentemente funcionan con poco personal, y los empleados realizan diversos roles, por ejemplo: programadores, analistas de sistemas, líderes de proyecto entre otros. Aun así, los empleados deben tener las habilidades requeridas para ejecutar sus diversos roles, e incorporar procesos que sustenten la calidad del producto. Dada esta necesidad de conocimiento multidisciplinario, las universidades enfrentan retos importantes para formar profesionistas que puedan insertarse de una manera rápida y sin complicaciones al sector laboral. Según (Garg & Varma, 2008), la falta de profesionales adecuadamente formados es un obstáculo importante en el sustento y crecimiento de la industria de desarrollo de software. Entonces, la capacitación deberá incluir mecanismos que permiten guiar, entrenar y adaptar a los individuos dentro de una economía globalizada y que les permita ser más competitivos local e internacionalmente. Por otro lado, el proceso de capacitación en sí mismo puede verse como uno que requiere modificarse y perfeccionarse constantemente.

En esta investigación se propone un nuevo Modelo de mejora del proceso de capacitación para desarrolladores de software en PYMES basado en ambientes ágiles. El modelo propuesto, llamado CAAM (Modelo de Capacitación en Ambientes Ágiles bajo Moprosoft), se inspira en los modelos de mejora de procesos y en los modelos ágiles. CAAM es un modelo de capacitación que permite el avance continuo e incremental en las habilidades de los desarrolladores, y a su vez promueve un progreso constante y medible en la productividad de la empresa y en la calidad del software producido. Además, CAAM permite al capacitador incorporar técnicas de gestión de desarrollo ágil, como Scrum (Harrison & Foote, 2000), así como desarrollar bajo el modelo de mejora de procesos Moprosoft (Oktaba, Alquicira, Su, & Martínez, 2005). CAAM parte del supuesto de que la empresa involucrada ha sido verificada a menos en el nivel 1 de la norma mexicana de calidad NMX-I-059-NYCE-2005 (Normalización y Certificación Electrónica A.C., 2007); (Flores B. , Astorga, Olguín, & Andrade, 2008). Este nivel, aunque aún está lejos de representar el ideal en cuanto a madurez de desarrollo del proceso de software, requiere al menos del uso de componentes básicos y sistematizados de desarrollo, lo que otorga una pauta de arranque a CAAM.

1.1 Planteamiento del problema

La ingeniería de software incluye modelos, estándares y métodos altamente probados experimentalmente y que el profesionista puede aplicar. Sin embargo, resulta muy frustrante cuando recién graduados y/o desarrolladores con experiencia no consiguen adaptar estos modelos a las necesidades de la empresa y en consecuencia se dificulta enormemente su inserción en el ámbito laboral. Aún así, en la actualidad es solicitado el uso de estándares y modelos de desarrollo por muchas compañías, tanto a nivel nacional como internacional. Esto implica que las PyMEs deben erogar en gastos de capacitación, pues muchas veces este conocimiento no está incluido dentro de los planes y estudios universitarios, dando pauta a que sus egresados carezcan de disciplina en el uso de los mismos. Si la prioridad principal de

las PyMEs es prevalecer en el mercado en vez de establecer un modelo de desarrollo, entonces se dificulta fuertemente su incorporación en ambientes competitivos.

El combinar metodologías ágiles con modelos de mejora de procesos, tiene como objetivo adoptar las ventajas de ambos enfoques. Siguiendo este enfoque, es posible utilizar un esquema de entrenamiento que haga más natural la adopción de estos esquemas por parte de los integrantes de los equipos de desarrollo.

1.2 Propósito de la investigación

Se propone en el presente trabajo que las PyMEs que tengan al menos el nivel 1 de la norma NMX-I-059-NYCE-2005 adopten un Modelo de mejora del proceso de capacitación para desarrolladores de software basado en ambientes ágiles, medible para que los integrantes del desarrollo se disciplinen en ambientes de trabajo con filosofías de desarrollo ágil. Con esto se espera que las PyMEs logren incrementar su productividad, y mantener sus proyectos dentro de un presupuesto y tiempo de desarrollo planeados. A través de esta investigación, se han definido estrategias que permiten disciplinar a los integrantes de los equipos de desarrollo en el uso de modelos y herramientas ágiles de manera incremental.

1.3 Objetivo general

Desarrollar un modelo de mejora del proceso de capacitación para desarrolladores de software en PYMES basado en ambientes ágiles, verificados en el nivel 1 de Moprosoft, que les permita un progreso constante y medible en las capacidades de los desarrolladores de software y en la calidad del software producido.

1.4 Objetivos específicos

1. Diseñar un modelo de mejora de procesos de capacitación que cumpla con los requerimientos de los modelos de mejora y se base en métodos ágiles.
2. Identificar un proceso de medición de las capacidades individual y en equipo aplicable a los métodos presentados en el modelo de mejora.

1.5 Aportación

La principal aportación de esta investigación es un nuevo modelo que permite seguir procesos de mejora en la capacitación de profesionistas desarrolladores de software que cuentan con poco tiempo para realizar esta actividad. El modelo supone que la empresa tiene el nivel básico de sistematización (Nivel 1 de NMX-I-059-NYCE-2005) y que el desarrollo de software se realiza a través de un método ágil.

1.6 Justificación de la investigación

En esta tesis se aporta un modelo de mejora en el proceso de capacitación para las PyMEs que tengan al menos el nivel 1 de la norma NMX-I-059-NYCE-2005, para que los integrantes del desarrollo se disciplinen en ambientes de trabajo con filosofías de desarrollo ágil cuidando la arquitectura de Moprosoft nivel 1. El combinar las metodologías ágiles con modelos de mejora de procesos, tiene como objetivo adoptar las ventajas de ambos enfoques. Esta amalgama permitirá generar una sinergia que resulte en beneficio del desarrollo de software, ya que el proceso ágil permitirá la adaptación a las necesidades competitivas actuales aún y cuando se estén realizando pequeños proyectos sin perder las ventajas metodológicas que están presentes en la administración y control de proyectos grandes. Siguiendo este enfoque, se podría utilizar un esquema de entrenamiento que haga más natural la adopción de estos esquemas por parte de los integrantes de los equipos de desarrollo.

1.7 Alcances y limitaciones

Con respecto a los alcances, se espera que el uso de CAAM reduzca el tiempo de adopción de los procesos en las pequeñas y medianas empresas de desarrollo originando un ahorro en los gastos de implementación. Asimismo, las estrategias de capacitación propuestas permitirán estandarizar las capacidades y habilidades en las empresas con nivel 1 de NMX-I-059-NYCE-2005 para incorporar prácticas ágiles en el desarrollo dedicando no más de 40 horas anuales en el proceso.

Con respecto a las limitaciones de CAAM, éste se enfoca a una técnica ágil de desarrollo, basada en Desarrollo Dirigido por Pruebas. Se seleccionó este modelo

debido a que su naturaleza permite generar pequeños fragmentos de código y permite llevar el concepto de refactorización hasta obtener un diseño final óptimo, características que se consideran importantes en el contexto en que se realiza la investigación. Por otra parte, el desarrollo de esta investigación se acotó a un tiempo no mayor a dos años, evaluándola a través de la aplicación del modelo en profesionistas del desarrollo de software graduados, con y sin experiencia en la aplicación de modelos de mejora de procesos. Por esta razón, no se incluye un estudio de campo de varios años.

1.8 Descripción del documento

Este documento está formado de la siguiente manera: en el capítulo 2 se presenta el marco teórico necesario para entender el resto de la tesis y describe brevemente los trabajos más relevantes dentro de esta área de investigación. En el capítulo 3 se describe la metodología utilizada para llevar a cabo la investigación. En el capítulo 4 se describe a detalle el modelo propuesto. En el capítulo 5 se discuten y analizan los resultados obtenidos al aplicar el modelo en varios experimentos y en el capítulo 6 se resumen las conclusiones y se propone trabajo a futuro a realizarse sobre este mismo tema.

Capítulo 2 MARCO TEORICO

En este capítulo se resumen los conceptos principales asociados a tres ejes de conocimiento relacionados a esta investigación: los modelos de mejora de procesos como es Moprosoft, la capacitación empresarial, y los modelos ágiles. Asimismo, se incluye una sección con la revisión de los principales trabajos recientes publicados relacionados con este tema.

2.1 Modelos de mejora de procesos

Se define a un proceso como un conjunto de tareas que, ejecutadas ordenadamente, producen los resultados deseados (Humphrey, 1989). Existen algunas

ligeras variantes en la definición de este término cuando se relaciona al desarrollo de software, por ejemplo:

- IEEE: “Una secuencia de pasos realizados para un fin determinado” (IEEE1074-97, 1997).
- CMMI 3,4: “Consiste en actividades que pueden ser reconocidas como las implementaciones de las prácticas en un modelo CMMI” (Glacer, 2008).
- PMBOK: “Una serie de acciones para lograr un resultado... y de esta manera mantenerse en la vanguardia que demandan los mercados globalizados” (PMI, 2004).

El establecer procesos o mejorar los actuales es de suma importancia en cualquier área de una organización. La Fig. 2.1 se resume las etapas de adopción de procesos, sugeridas por (Humphrey, 1989). En el afán de dar mejores resultados se han generado varios enfoques como la gestión de calidad total (Hammer M. & Champy J., 1993), la reingeniería de procesos y el aprendizaje organizacional (Checkland P. & Scholer J., 1990).

Los modelos de mejora de procesos son de vital importancia en las empresas desarrolladoras de software. El propósito fundamental de los modelos de mejora es guiar a las empresas a la madurez y capacidad de sus procesos actuales. Esta madurez debe llevarse a cabo de manera incremental y controlada.

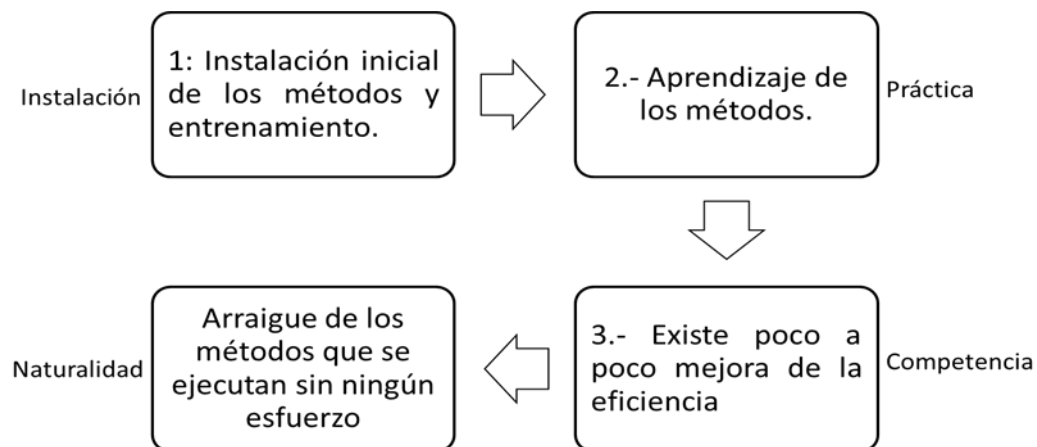


Fig. 2-1 Etapas de adopción de procesos (Humphrey, 1989).

2.1.1 Modelos de mejora de procesos de software

Los procesos de software están formados por un conjunto de tareas, métodos y prácticas usados para tener mejores resultados en sus productos, es decir, entregas dentro de agendas, evitar las pérdidas económicas y principalmente la mejora de la calidad. El objetivo de los procesos de software es administrar los productos de acuerdo a un plan, mientras simultáneamente se logra la capacidad de producir mejor. El paso más importante para solucionar los problemas asociados a producir software, es tratar a todas las tareas como un proceso que puede ser controlado, medido y mejorado. Desde hace varios años, es muy común utilizar la estructura propuesta por Humphrey, a fin de establecer la posición de la organización con respecto a sus procesos (Humphrey, 1989). La fig. 2.2 muestra esta estructura, cuyo principal objetivo es lograr una constante mejora en los procesos. Asimismo, se cuenta con un mecanismo de evaluación que ayuda a identificar el estado de cada uno de los niveles para el aseguramiento del cumplimiento del mismo. La mejora de procesos es una cultura que puede ser vista de manera internacional o nacional, como lo muestran diferentes modelos que existen actualmente: SW-CMM, CMMi, PSP, TSP, ISO/IEC 15504, PMBOK y SWEBOK (Ruvalcaba, 2005).

Los modelos de calidad de procesos de software fueron desarrollados en el contexto de las grandes organizaciones y empresas multinacionales. De hecho, existe una tendencia generalizada a suponer que el éxito de la mejora de procesos de software sólo es posible para las grandes empresas que disponen de recursos suficientes para hacer frente a estos tipos de prácticas. Esta percepción se basa en el hecho de que los programas de mejora de procesos simplemente no son viables para las PyMEs, debido a su estructura organizacional y los altos costos que esto involucra (Hareton & Terence, 2001). Sin embargo, la industria del software en la mayoría de países se compone principalmente de PyMEs, que favorecen el crecimiento de las economías nacionales (Conradi & Fuggeta, 2002). En algunos países de Latinoamérica la mayoría de las organizaciones son pequeñas, con un comportamiento indisciplinado, con procesos *ad-hoc* y falta de concentración en el proceso. Desgraciadamente, es un hecho que las PyMEs se centran principalmente en

el producto más que en el proceso, lo que lleva en ocasiones al desconocimiento de los modelos de desarrollo. Algunas propuestas de investigación como la de Centeno y Gómez-Gil proponen una manera original de incorporar la mejora de procesos para las PyMEs dedicadas al desarrollo de software (Centeno-Tellez & Gómez-Gil, 2010).

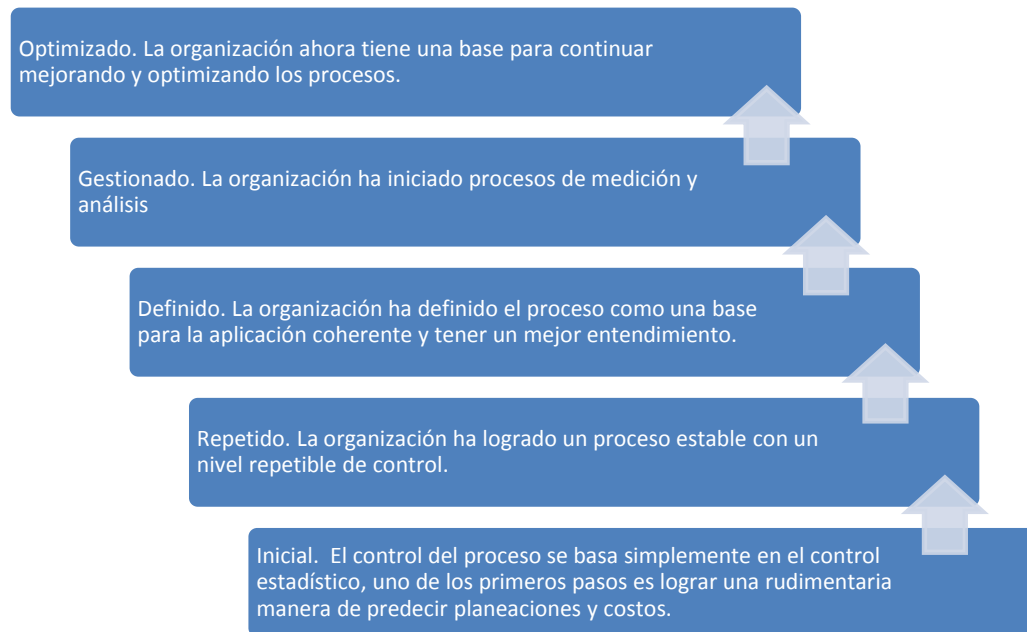


Fig. 2-2 Niveles de madurez de procesos (Humphrey, 1989).

La actividad de mejora de procesos de software (conocida como SPI, por sus siglas en inglés) muestra un enfoque gerencial en lugar de tratar directamente con las técnicas de desarrollo. Algunos de modelos actuales de SPI son:

- *Modelo de Procesos para la Industria de Software (Moprosoft)*, categoría de operación, (Oktaba, Alquicira, Su, & Martínez, 2005);
- *Software Engineering Institute's (SEI) Capability Maturity Model (CMM)*, both the *Software CMM (SW-CMM®)* and *CMM Integration (CMMI)* (CMMI, 2010);
- *Software Process Improvement and Capability Determination (SPICE)* (SPICE, 2011)
- *Total Quality Management (TQM)* (Martínez, Dewhurst, & Dale, 1988);

- *Software Process Improvement in Regions of Europe* (SPIRE, 1999);
- *Project Management Institute's (PMI) Project Management Body of Knowledge* (Nokes & Kelly, 2007);
- *Six Sigma* (Escalante & Seis, 2003).

Existe una gran cantidad de estudios relacionados con la mejora de procesos, por ejemplo los desarrollados por (Dueholm, 2010), (Niazi, 2005). Niazi realizó una investigación de SPI y obtuvo de la revisión de 47 artículos que un 49% de los profesionistas opinan que, para conseguir una mejora al ejecutar procesos, es necesario el entrenamiento y la tutoría. De la realización de entrevistas a 23 profesionales obtuvo que un 65% afirman que el éxito de un programa de implementación de SPI depende de entrenamiento y tutoría. Entre los principales modelos de SPI que han aportado en diversas áreas tanto de la industria como para la academia está el modelo CMMI.

2.1.2 Modelo CMMI

CMMI (Capability Maturity Model Integration) es un modelo de madurez de mejora de procesos para el desarrollo de productos y servicios (Beth, Konrad, & Shrum, 2009). Consiste en las mejores prácticas que tratan las actividades de desarrollo y de mantenimiento que cubren el ciclo de vida del producto, desde la concepción a la entrega y el mantenimiento. Este modelo tiene su origen de la combinación de diversos esquemas (SW-CMM, SECM, IPD-CMM) con la finalidad de permitir a las organizaciones tener una mayor mejora en sus procesos en toda su empresa. Debido a sus orígenes del modelo se puede adoptar en diversos marcos de mejora de diversos dominios. El CMMI le permite aproximarse a la mejora de procesos y a la evaluación usando dos niveles:

- Los niveles de capacidad, que pertenecen a la representación continua, se aplican al logro de mejora de procesos de una organización en áreas de proceso individuales. Estos niveles son un medio para mejorar de forma incremental los procesos que corresponden a un área de proceso dada.

- Los niveles de madurez, que pertenecen a la representación por etapas, se aplican al logro de mejora de procesos de una organización en múltiples áreas de proceso. Estos niveles son un medio de predecir los resultados generales del siguiente proyecto que se acometa.

Tanto los niveles de capacidad como los niveles de madurez proporcionan una forma de medir cómo las organizaciones pueden y hacen mejorar sus procesos. La representación continua permite a la organización elegir el enfoque de sus esfuerzos de mejora de procesos mediante la elección de aquellas áreas de proceso interrelacionadas, que beneficien más a la organización y a sus objetivos de negocio. En esta fase se puede madurar de manera separada cada área de proceso, por ejemplo a una organización puede desear esforzarse por alcanzar el nivel de capacidad 2 en un área de proceso y el nivel 4 en otro. A medida que la organización alcanza un nivel de capacidad, establece su visión para el siguiente nivel de capacidad para una de estas mismas áreas de proceso o decide extender su visión y tratar un número más grande de áreas de proceso. Ambos enfoques hacen uso del concepto de áreas de procesos que dentro de CMMI es el punto central a tomar en cuenta para la madurez o capacidad de la organización éste concepto tiene una estructura para llevar a cabo una dirección de las actividades a desarrollar.

2.1.3 Patrón del área de procesos de CMMI

Se define un área de proceso como un grupo de prácticas relacionadas en un área que, cuando se implementa de forma conjunta, satisfacen un objetivo considerado como importante para la mejora en esa área (Beth, Konrad, & Shrum, 2009). Para el diseño del modelo de mejora propuesto en esta investigación, se toman algunos elementos de modelo propuesto en CMMI ver 1.3, los cuales se describen a continuación:

Notas introductorias. La sección de “notas introductorias” del área de proceso describe los conceptos principales cubiertos por el área de procesos y es un componente informativo.

Áreas de proceso relacionadas. Es una lista de referencias a áreas de proceso que están en relación y refleja las relaciones de alto nivel entre las áreas de proceso.

Metas específicas. Describen las características únicas que deben de satisfacer el área de proceso. Una meta específica es un componente requerido del modelo que se utiliza en las evaluaciones para ayudar a determinar si se satisface un área de proceso.

Metas genéricas. Describe las características que deben estar presentes para institucionalizar los procesos que implementan un área de proceso.

Prácticas específicas. Es la especificación de una actividad que se considera importante para alcanzar la meta específica asociada. Las prácticas específicas describen las actividades que se esperan que produzcan la consecución de las metas específicas de un área de proceso.

Prácticas genéricas. Se denominan así porque las mismas se aplican a múltiples áreas de proceso. Una práctica genérica es la descripción de una actividad que se considera importante para el logro de la meta asociada.

2.2 Capacitación en la ingeniería de software

Existe una importante variedad en métodos para la capacitación de ingeniería de software (Steve D., 2010). Algunas técnicas se ejecutan en ambientes de desarrollo reales, donde los recién graduados o los estudiantes de los últimos niveles de su preparación de pregrado, pueden capacitarse de manera previa antes de enfrentarse a los ambientes reales de trabajo (Dawson, 2000). Otro esquema de entrenamiento implica el uso de sistemas diseñados para proporcionar entrenamiento basándose en el uso de la computadora, permitiendo avanzar al siguiente nivel siempre y cuando se acredite en el que se está.

Sea cual fuere el estilo utilizado, la capacitación se debe considerar como un intento de mejorar los procesos del trabajo en lugar de un requisito previo para el desempeño exitoso en la parte técnica. Asimismo, debe recordarse que el motivo de la implementación de un mecanismo de capacitación en el área de software es el generar competencias en los procesos de madurez y no tanto en el aspecto técnico. Aunado a

lo anterior, los costos generados por la implantación de cursos debe de disminuir (Garg & Varma, 2008) a fin de ser más accesible el uso de la mejora de procesos de calidad en las empresas desarrolladoras que tienen escasos presupuestos.

Es de vital importancia tener un objetivo claro para diseñar un plan de capacitación. Debe enfocarse principalmente en que los individuos sepan operar y tengan las habilidades para ejecutar correctamente sus actividades, lo que implica que conlleven sus roles y procesos para mejorar continuamente en el trabajo que están haciendo (Cuellar C., 2010). Esto implica el contar con elementos que permitan mantener actualizados los procesos que ya se llevan a cabo, además de incluir nuevos, lo que permitirá la madurez. Desgraciadamente existe una disparidad de lo que se muestra en los procesos de capacitación con lo que realmente se enseña en las universidades. Los materiales de enseñanza se quedan atrás debido en muchas ocasiones a la falta de flexibilidad de los planes de estudio (Huang P., Chen M. & Chen S., 2008); (Li, Liu, & Guo, 2008).

La dinámica del desarrollo de software es cambiante, por lo tanto la capacitación es necesaria para mantener el ritmo en la obtención de los conocimientos adecuados como se propone en el modelo de entrada-proceso-salida propuesto en (Bushnell, 1990). Asimismo, la elaboración de software requiere de un proceso de capacitación que de manera previa estandarice los diversos escenarios donde se ubican los empleados de mayor edad (Sterns & Doverspike, 1989).

(Rogers, Campbell, & Pak, 2001) señalan que es fundamental un uso coherente de las directrices con las que se cuentan para la generación de un proceso de capacitación. Dicho autor sugiere lo siguiente:

1. Las herramientas educativas que conforman el proceso de capacitación deben estar en equilibrio con los procesos cognitivos, físicos o psicomotores que conducen al dominio requerido por la institución involucrada.
2. El alumno debe ser inducido para desarrollar su capacidad, es decir, tener una mayor actividad de memorización y transferencia del conocimiento adquirido.

3. Todas las fuentes disponibles de comentarios se deben de utilizar y la retroalimentación debe ser precisa, válida y constructiva.
4. Los procesos de capacitación deben de mejorar la práctica para obtener un mejor resultado.
5. Los procesos de capacitación deberán ser adaptados a diferentes aptitudes, prácticas y conocimientos previos.

En una encuesta realizada por (Mejía, Ania, & Gamboa, 2006) se considera que hay escasez de recursos humanos, principalmente de aquellos con competencias para el aseguramiento de la calidad, la alta dirección y la administración de proyectos, más que con competencias para el desarrollo de los proyectos en sí. Por otra parte, la realidad actual es que el proceso de capacitación en los desarrolladores de software es demasiado largo, incompatible muchas veces con las urgencias de entrega que provocan los desarrollos de software. Varios administradores de desarrollo y compañías de consultoría están intentando acelerar estos procesos. Por ejemplo, se cuenta con un método que permite reducir el nivel de expertos de desarrollo de software de 2 años a 4 o 6 meses, mediante el proceso de compartición de conocimiento (Bologa & Lupu, 2007).

Los factores usados para medir la efectividad de una actividad de capacitación, pueden estar dados por las categorías siguientes: cambios en el comportamiento, cambios en la actitud, adquisición de conocimiento y habilidades, medición de resultados tangibles, reacción y participación (Kirkpatrick D., 2009). Desde esta perspectiva, es posible distinguir al menos cuatro niveles de evaluación de la capacitación, basándose en la recopilación y análisis de datos (Goldstein, 1993):

1. Evaluación del proceso de capacitación, la cual está dirigida a verificar si se cumplió en la práctica el diseño de instrucción previsto. Deben considerarse aquellos aspectos que influyen en la calidad y la eficiencia del proceso de capacitación.
2. Evaluación del aprendizaje, el cual se entiende como la adquisición de competencias técnicas o psicosociales mediante experiencias sistematizadas con

objetivos de aprendizaje medibles. En este punto se deben tener en cuenta las competencias a evaluar, las cuales deberán ser:

- a) Competencias técnicas
 - i. Conocimientos
 - ii. Habilidades intelectuales
 - b) Destrezas sicomotoras
 - i. Competencias psicosociales
 - c) Habilidades interpersonales
 - d) Disposición anímica en el trabajo
3. Evaluación del impacto de la capacitación en el desempeño laboral. Tiene por objetivo comprobar en qué medida las competencias aprendidas se reflejan en mejoras en el desempeño del trabajador. Se entiende por desempeño el comportamiento del trabajador en el marco de las normas y objetivos de la organización. Para esto se puede hacer un muestreo estadístico antes de realizar la capacitación utilizando por ejemplo, el porcentaje de los participantes no capacitados.
4. Evaluación de la rentabilidad de la capacitación. Ésta tiene por objeto determinar en qué medida los resultados de la capacitación compensaron sus costos, en cuanto a mejoras de la eficiencia o productividad en el trabajo o del clima organizacional en la empresa. Se trata, pues, de una evaluación de beneficios y costos realizada desde la perspectiva de la gerencia de la empresa.

2.2.1 Un modelo sistemático de capacitación

Para desarrollar esta investigación, se realizó un análisis de diversos trabajos de capacitación propuestos en la literatura (Bushnell, 1990); (Barbazette, 2006). De particular interés resulta el modelo propuesto por (Buckley R. Caple J., 2009), ya que representa un enfoque sistemático de entrenamiento, identificando los elementos para el funcionamiento de un programa de capacitación genérico (ver fig. 2.3). En el modelo se incluyen, a grandes rasgos, características de conocimiento, habilidad y

desarrollo de aptitudes, mismas que forman parte de una capacitación basada en competencias. Enseguida se describe brevemente cada componente:

- 1 *Términos de referencia.* Antes de iniciar la capacitación se realiza un plan de acción que deberá ser elaborado por el entrenador, dentro de un límite de tiempo razonable a fin de que los clientes puedan mantenerse involucrados.
- 2 *Análisis adicional.* Se incluyen procesos que ayuden a identificar a la capacitación como una solución a problemas actuales o problemas potenciales.
- 3 *Conocimientos, habilidades y análisis de actitud.* Es la definición de conocimientos, habilidades y actitudes actuales del individuo, que permitirá proceder a realizar la toma de decisiones apropiadas sobre la naturaleza y el tipo de capacitación necesaria.
- 4 *Análisis de la población destino.* Se evalúan capacidades para determinar otras características de la función objetivo, la cual define el motivo de la capacitación. Esto puede incluir un examen de la forma y eficacia de los programas actuales.
- 5 *Necesidades de capacitación y análisis de contenido.* Se deben obtener datos personales acerca de los trabajos y tareas necesarias para la capacitación. Esto permite establecer los términos de referencia de lo que se necesita aprender y el grado de entrenamiento requerido.
- 6 *Desarrollo de criterios de medición.* Es necesario establecer un estándar o nivel que mida el rendimiento y de esta manera poder realizar un cambio de nivel o progreso de la capacitación, si se amerita. Estos criterios deben ser considerados durante las primeras etapas de la investigación y serán incorporados en los objetivos, antes que cualquier estrategia de formación pueda ser considerada.
- 7 *Preparación de objetivos de entrenamiento.* Los objetivos de formación deben ser por escrito para describir con precisión lo que se espera que sean capaces de hacer los alumnos, como resultado de su experiencia de aprendizaje. Normalmente, en el objetivo se presentan tres partes. En la primera se describen las habilidades y conocimientos que se adquieren. La segunda parte describe las condiciones de trabajo. La tercera parte establece qué se espera lograr al final de la capacitación

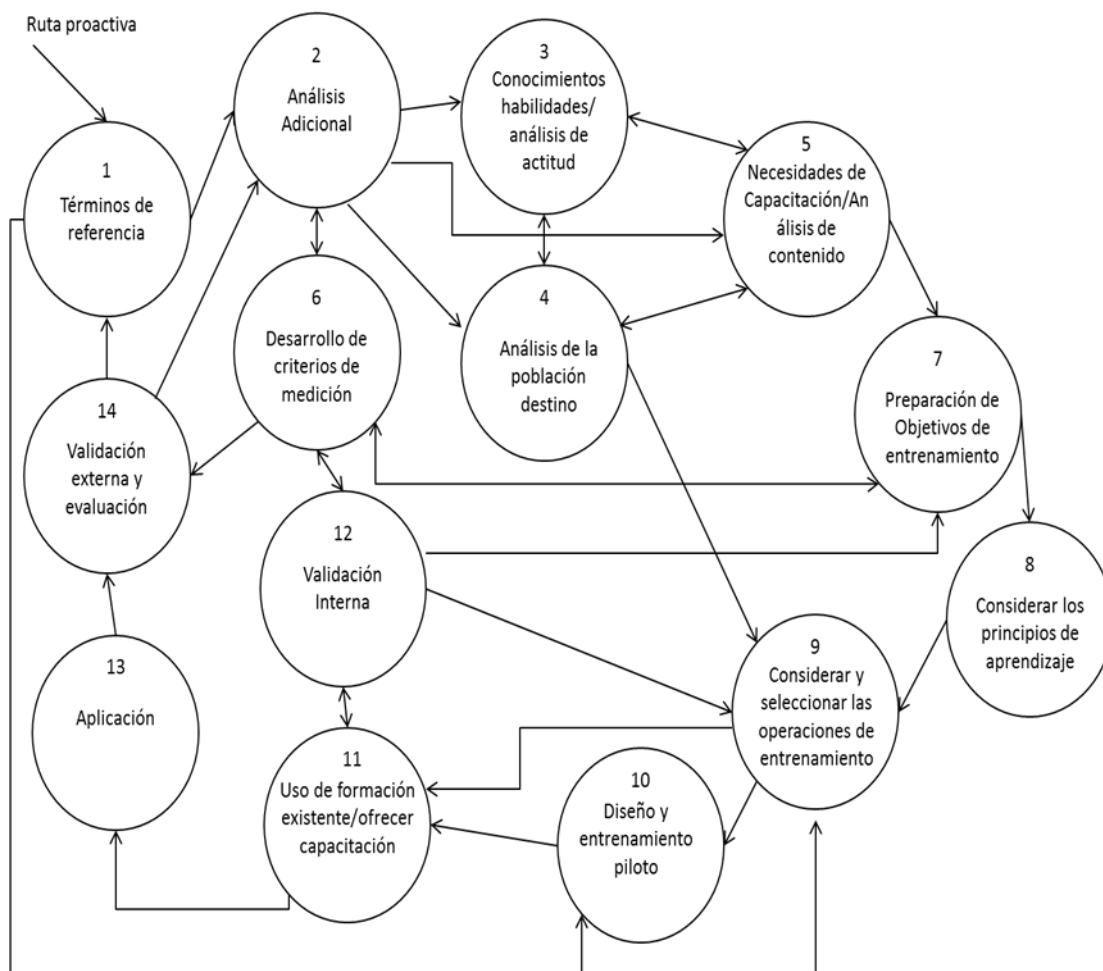


Fig. 2-3 Fases de un diseño de capacitación (Buckley & Caple, 2009).

8 *Considerar los principios de aprendizaje.* Es necesario tener en cuenta los principios del aprendizaje y la motivación. Además, los alumnos necesitan desarrollarse en un medio ambiente que asegure que los objetivos de la capacitación se puedan lograr.

9 *Considerar y seleccionar las operaciones de entrenamiento.* Diferentes opciones están disponibles para el entrenador a la hora de seleccionar la forma más adecuada de la capacitación. Deben considerarse las diversas formas de aprendizaje, los cursos abiertos, en línea, o fuera de la localidad laboral, metodologías de entrenamiento, entre otras.

10 *Diseño y entrenamiento piloto.* El diseño de la capacitación consiste en la traducción de los objetivos y estrategias en un programa equilibrado de enseñanza y aprendizaje. Esto no necesariamente significa un curso, sino que podría ser un paquete de aprendizaje, un vídeo, el entrenamiento basado en computadoras, diferentes métodos y estrategias de capacitación.

11 *Uso de formación existente y ofrecer la capacitación.* Impartir la capacitación involucra varias técnicas a desarrollar, donde el capacitador deberá de manejar los conocimientos técnicos de una manera competente.

12 *Validación Interna.* Es el proceso de medir el desempeño de los alumnos para ver si han logrado los objetivos de la formación. Esto puede hacerse de dos maneras. La primera involucra una serie de pruebas, ejercicios e instrumentos de evaluación que son diseñados y utilizados para examinar objetivamente o para comprobar el progreso de los alumnos. La segunda implica que los entrenadores necesitan conocer los puntos de vista de los alumnos, incluyendo factores tales como el rendimiento de sus tutores o instructores, los materiales de aprendizaje y el medio ambiente.

13. y 14. *Aplicación y control externo de la capacitación.* Una vez que la capacitación ha tenido lugar los alumnos deben aplicar sus conocimientos y habilidades en el trabajo, es decir, desarrollar las competencias adquiridas. Cuando los ex-alumnos han tenido oportunidad suficiente para poner en práctica lo que han aprendido durante su capacitación, el proceso de control externo debe ser introducido. Este proceso implica la validación externa y la evaluación de la capacitación. La información obtenida en este modelo sirve como base para posibles modificaciones a la metodología y a los contenidos de la capacitación. Además este proceso permite medir el valor total de la formación en los beneficios individuales, de equipo y operativos.

2.3 Métodos ágiles

Los métodos ágiles son un paradigma en el desarrollo de software que se originan de la necesidad de reducción de riesgos en la entrega. Esto implica el obtener

tiempos de entrega de acuerdo a lo planeado, trabajar en una manera colaborativa con los *stakeholders* para alcanzar un fin común y conseguir calidad en el desarrollo. Por otra parte, involucran al cliente en el desarrollo, a fin de asegurar que todos los requerimientos son levantados y que con esto se reduzcan riesgos y se aumente la precisión siguiendo el ciclo de desarrollo ágil: asegurar-adaptar-actuar (VisionOne, 2010). La naturaleza iterativa de entregables de los métodos ágiles y la constante comunicación con el usuario mantienen un dinamismo en el desarrollo de software, a diferencia de lo ocurrido durante el desarrollo tradicional. La habilidad de entregar el software de manera gradual permite una mejor administración en cada etapa, asegurando la entrega de dos a tres semanas o en su caso una redefinición de los tiempos.

Los principios del desarrollo ágil incluyen: definir equipos divididos en pequeñas piezas de trabajo, hacer entregas de desarrollo en tiempos cortos y realizar iteraciones frecuentes. El éxito de los equipos ágiles gira alrededor de un compromiso de trabajar en equipo, constante comunicación para percatarse de los cambios y actuación rápida. Las principales ideas detrás del desarrollo de software ágil son: primero implementar el manifiesto ágil (Cockburn, Agile Software Development, 2000) y segundo presentar determinadas prácticas que puedan ser habilitadas en los equipos para obtener una mejor calidad en el mismo (Cockburn, Agile Software Development, 2000). Algunas de estas prácticas son: Programación Extrema (Dubinsky & Hazzan, 2003), *Scrum* (Schwaber & Beedle, 2001), Desarrollo de Software Adaptativo (DSDM por sus siglas en inglés) (Stapleton, 1997), *Crystal* (Cockburn, 2005) y Desarrollo Dirigido por Pruebas (TDD, por sus siglas en inglés) (Beck E. , 2003).

Es importante resaltar que una metodología ágil se enfoca en el desarrollo del individuo más que en el proceso. Además, el involucramiento del usuario es fundamental, haciéndolo tomar decisiones y respondiendo a cambios de manera oportuna con compromiso y calidad. El manifiesto ágil, publicado por (Beck, Beedle, Bennekum van, & Cockburn, 2001) determina: tener equipos divididos en pequeños grupos de trabajo, entregas de desarrollo en tiempos cortos, iteraciones

frecuentes que produzcan entregas funcionales rápidas y prioridad en la comunicación en todos los sentidos y durante todo el proceso.

2.3.1 Desarrollo dirigido por pruebas

La importancia de la escritura de pruebas radica en la posibilidad de reducir código, una cohesión entre los métodos de las clases y una continua reconstrucción, lo que permite una optimización en la fase de construcción del software. Un código bien escrito afecta positivamente en los costos de mantenimiento ya que permite una disminución en los tiempos de revisión, una fácil comprensión de lo que se escribe y aminora la complejidad ciclomática¹ (McCabe T., 1976). El beneficio obtenido a nivel grupo es lograr que los programadores puedan realizar los diseños durante el desarrollo. Varias empresas de desarrollo de software han adoptado la filosofía de “probar primero y luego codificar” generando automáticamente las pruebas mediante marcos de desarrollo como *xUnit* (*JUnit* o *NUnit*) (Nagappa & Maximilie, 2008).

En el proceso de TDD, la prueba es ejecutada para validar fallas antes de iniciar la codificación. Este proceso ayuda a implementar a la par las pruebas de integración, que históricamente se han realizado poco durante el desarrollo (Sull F., Melnik G., Turhan B., Layman L., Diep M. and Erdogmus H., 2010). En muchas ocasiones el enfoque que se le da a TDD es erróneo, ya que el nombre de *Test*, lo orienta a considerarse solamente en las etapas de verificación y validación, pero realmente el sentido de esto es la mejora en el diseño. Varias investigaciones soportan esta idea, en las cuales se tienen el estudio realizado por (Janzen D., 2008) se encontró que, aplicando pruebas al finalizar la codificación, el número de bifurcaciones lógicas en un programa era de un 74%; al aplicar pruebas al inicio se alcanzó un 94% de mejora. Otro beneficio observado en el mismo estudio es la personalización de la técnica por parte de las personas que la aprenden. Esto genera una tendencia en los programadores que utilizan TDD, para escribir clases y métodos con una mayor sencillez y con un mínimo de código. Además el estudio enuncia que

¹ Medición indirecta que se calcula a partir de varios indicadores, entre los que se encuentra el número de ramificaciones que tiene el código. Se usa como parámetro para evaluar la calidad del software o del diseño, la legibilidad, la comprensibilidad y facilidad de mantenimiento.

los estudiantes con menos experiencia son más abiertos a la adopción al enfoque de probar primero que los estudiantes que fueron entrenados por un corto tiempo con el enfoque de “pruebas al final.”

Enseñar diseño y pruebas de software puede ser especialmente difícil (Janzen D., 2008). Los cursos de licenciatura y los programas de entrenamiento en las industrias relegan a estas dos actividades, haciendo pensar que éstas son difíciles de implementar e incluso opcionales de llevar a cabo. Janzen propone una filosofía de enseñanza para aprender el uso de las pruebas basándose en la idea de si el objetivo es escribir buenas pruebas, entonces se debe por comenzar a enseñar como generar pruebas; si el diseño del software es la meta, entonces iniciar por enfocarse en los buenos hábitos para un buen diseño, por ejemplo unidades de código más pequeños y mejor cohesionados ².

2.3.2 Scrum

Scrum es un proceso de desarrollo de software para equipos pequeños. En diversos estudios se demuestra que la efectividad de los trabajos depende de los equipos pequeños, donde cada miembro entiende su función, rol y que entregable debe de proporcionar en avances cortos e iterativos dentro de un proyecto (Rising & Janoff, 2000). Para aplicar *Scrum* y aprovechar su flexibilidad, hay que utilizar las siguientes técnicas: “Creación del Equipo”, “Definición de los *Backlog*”³, “Segmentación del Proyecto” y “Reuniones”. En el proceso se conjugan diversos elementos, los que permiten la iteración e incremento en los productos que se desarrollan. Se puede realizar la combinación de *Scrum* con otras técnicas ágiles compatibles entre sí, por

² Es la manera en cómo se agrupan los métodos en una clase.

³ Definen los trabajos que hay que hacer. Hay 3 diferentes tipologías: los *Backlogs* del producto, que son los requisitos a cumplir en una dada versión del producto; los *Backlogs* de la release que son los requisitos elegidos para cumplir en una dada *release* del producto y finalmente los *Sprint Backlogs* que son los requisitos a realizar en una unidad de trabajo de máximo 4 semanas; mas cada elemento atómico del *Sprint Backlog* necesita 8-16 horas de trabajo (Albaladejo, 2010).

ejemplo, utilizar los casos de la programación extrema como elemento de refinamiento de código. Los equipos de *Scrum* están diseñados para optimizar la productividad, para lo cual, se auto-organizan, son multi-funcionales y trabajan en las iteraciones. Cada equipo *Scrum* tiene tres roles:

- 1) El *ScrumMaster*, que es el responsable de asegurar que el proceso se entienda y darle seguimiento;
- 2) El propietario del producto (*Product Owner*), responsable de maximizar el valor del trabajo que el equipo hace;
- 3) El equipo, formado por los que hacen el trabajo. El equipo se compone de los desarrolladores con todas las habilidades para convertir los requerimientos del propietario del producto en una pieza potencialmente entregable al final del *Sprint*.

Los elementos de *Scrum* incluyen una reunión de planificación de lanzamiento, otra de planificación un *Sprint*, *DailyScrum*, *Sprint Review*, y la *Sprint Retrospective*. El elemento principal de *Scrum* es el *Sprint*, que es una iteración que dura un mes o menos, utilizan el mismo marco de *Scrum*, y ofrecen un incremento del producto al final que es potencialmente entregable. Estos comienza inmediatamente después del otro. *Scrum* emplea los siguientes artefactos:

- 1) El *Product Backlog*, es una lista priorizada de todas las necesidades del producto.
- 2) La *Sprint Backlog* es una lista de tareas que el equipo elabora en la reunión de planificación como plan para completar los objetivos/requisitos seleccionados para la iteración y se compromete a demostrar al cliente
- 3) Un gráfico de trabajo pendiente a lo largo del tiempo muestra la velocidad a la que se está completando los objetivos/requisitos. Permite extrapolar si el equipo podrá completar el trabajo en el tiempo estimado.

La intención de *Scrum* es la de maximizar el desarrollo corrigiendo problemas mediante la observación temprana de los errores, para lo que se tiene el siguiente ciclo de vida:

1. **Pre-Juego: Planeación.** Establecer la visión, definir expectativas y asegurarse la financiación. Las actividades son la escritura de la visión, el presupuesto, el registro de acumulación o retraso del producto inicial y los ítems estimados, así como la arquitectura de alto nivel, el diseño exploratorio y los prototipos.
2. **Pre-Juego: Montaje.** El propósito es identificar más requerimientos y priorizar las tareas para la primera iteración. Las actividades son planificación, diseño exploratorio y prototipos.
3. **Juego o Desarrollo.** El propósito es implementar un sistema listo para entrega en una serie de iteraciones de treinta días llamadas *Sprint*.
4. **Pos-Juego. Liberación.** El propósito es el despliegue operacional. Las actividades, documentación, entrenamiento, mercadeo y venta.

Existen variantes en el uso de *Scrum*, estas son básicamente vistas en la construcción de las aplicaciones, en donde pueden existir técnicas de construcción mediante programación extrema o bien el de TDD mostrándose en la figura 2.4.

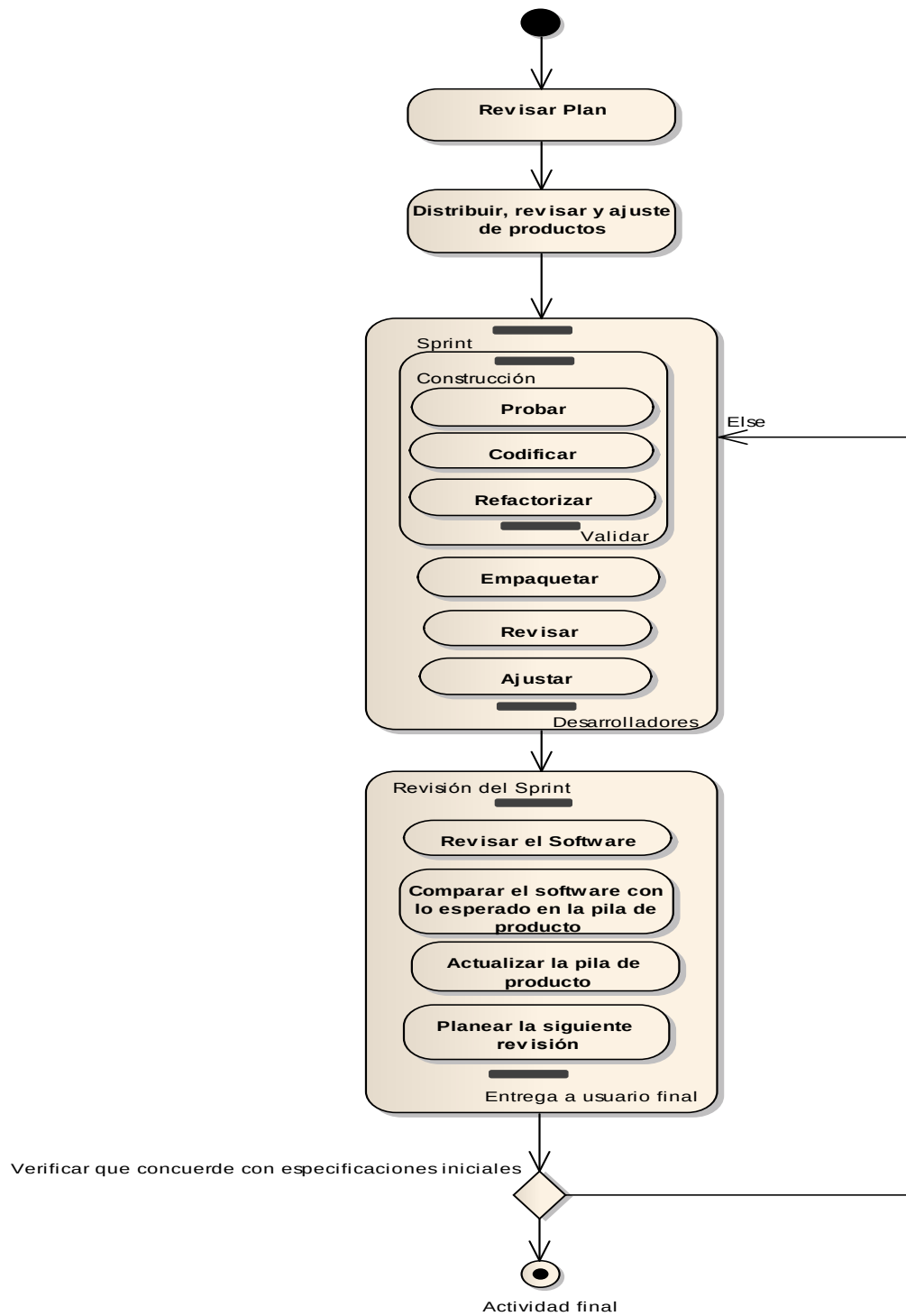


Fig. 2-4 Proceso de implementación de Scrum incluyendo TDD. Modificado de (Spada, 2005).

2.4 MoproSoft

Otro de los elementos a tratar en el presente trabajo es el “Modelo de Procesos para la Industria de Software” conocido como MoproSoft (Oktaba H. , García, Piattini, & Pino, 2006). Este modelo es uno de los resultados del “Programa para el Desarrollo de la Industria de Software” que lanzó la Secretaría de Economía en Octubre de 2002, con fin de crear las condiciones necesarias para que México cuente con una industria de software competitiva internacionalmente. MoProSoft está relacionado con ISO 9001:2000, CMM v1.1 e ISO/IEC TR 15504-2:1998. En México, la actividad exportadora se ha concentrado tradicionalmente en muy pocas empresas (Dedrick J., Kraemer K. & Palacios J., 2001). Las acciones de este programa “buscan” superar las limitaciones del sector de software implantando proyectos de trabajo en materia de: promoción externa, recursos humanos en cantidad y calidad suficientes, marco legal para las tecnologías de información y comunicación, desarrollo del mercado interno, fortalecimiento de la industria local, aplicación de modelos de evaluación y certificación de procesos e infraestructura adecuada para el sector” (Prosoft, 2002).

Moprosoft presenta una estructura para definir nuevos procesos y documentación de elementos, enfocándose a las pequeñas y medianas empresas o en pequeños grupos de desarrollo de software que formen parte de una organización más grande (Vargas, Olguín, & Flor, 2006). Existen varios casos de estudio de la aplicación del modelo. Por ejemplo, en (Flores B. , Astorga, Olguín, & Andrade, 2008) se siguió la siguiente metodología para su implementación: 1) identificación de factores de cambio que permiten la concientización del grupo de trabajo de la empresa; 2) el diagnóstico del estado real de la empresa; 3) aplicación de los 9 procesos obligatorios requeridos por el modelo, y 4) la evaluación oficial del cumplimiento de la capacidad de Moprosoft. En este proceso se identifican dos fases: primero una auto-evaluación detectándose áreas de oportunidad a cubrir, como fueron la mejora de los procesos en la gestión del negocio y la asignación de

responsabilidades en los 9 niveles que marca Moprosoft. En la segunda fase se encuentra la evaluación por NYCE de acuerdo a la norma NMX-I-059-NYCE-2005.

Se ha determinado que Moprosoft cubre en un 92% y presenta un alcance de 77% de las especificaciones generales y prácticas de CMMI en un nivel 2 (Rivera & Montero, 2004). Oktaba y colaboradores (Oktaba H. , García, Piattini, & Pino, 2006) llevaron un registro de la mejora en aquellas empresas mexicanas donde se ha aplicado el modelo, evaluando sus procesos mediante el uso de rangos, antes de adoptar a Moprosoft y después del entrenamiento en el mismo.

La incorporación de Moprosoft con métodos ágiles ha dado resultados a nivel Latinoamérica. Entre estos trabajos podemos encontrar a Competisoft (Oktaba H. , García, Piattini, & Ruiz, 2007) que tiene como objetivo el incrementar el nivel de competitividad de las PyMEs iberoamericanas productoras de software, mediante la creación y difusión de un marco metodológico común. La novedad en este modelo es la incorporación de la filosofía de métodos ágiles para la entrega de los productos al cliente, principalmente hace uso de *Scrum*. Según Glacer, el uso de los métodos ágiles y modelos de mejora puede crear actividades en los procesos más aceptables y adecuados (Glacer, 2008).

2.4.1 Estructura de procesos de Moprosoft

Para definir la estructura del modelo de proceso, en primer lugar se analiza la estructura de una empresa de desarrollo de software. Cabe hacer notar que, incluso una micro-empresa con menos de 10 personas, tiene un grupo de la alta dirección que toma las decisiones sobre la dirección del negocio. Además, se debe contar con un grupo de mandos intermedios que se encargue de la adquisición y control de los proyectos. Por último, debe existir un grupo de operaciones que desarrolla proyectos con los recursos asignados. Los miembros de esos grupos deben reconocer sus responsabilidades y realizar las funciones que les son asignadas. Los roles dan alineación de autoridad vertical y permiten una relación de colaboración horizontal entre ellos. En base a estas observaciones, Moprosoft divide sus procesos

en tres categorías: alta dirección, gestión y operación. El propósito de esta división es proporcionar procedimientos especializados para cada grupo funcional. La figura 2.5 presenta las categorías de MoProSoft y la estructura del proceso general mediante un diagrama de paquetes del lenguaje de modelado unificado (UML).

2.4.2 Categorías de Moprosoft

Las categorías de Moprosoft son las siguientes (Oktaba, Alquicira, Su, & Martínez, 2005, pág. 12). La categoría de Alta Dirección. Incluye los procesos que abordan las prácticas directivas importantes y fundamentales para el negocio, relacionadas con la gestión del negocio. Esta categoría proporciona los lineamientos a los procesos de la Categoría de Gerencia y se retroalimenta con la información generada por ellos. La categoría de Gerencia, incluye las prácticas de gestión de procesos, proyectos y recursos en función de los lineamientos establecidos en la Categoría de Alta Dirección. Además, esta categoría proporciona los elementos para el funcionamiento de los procesos de la Categoría de Operación, recibe y evalúa la información generada por éstos y comunica los resultados a la Categoría de Alta Dirección.

La Categoría de procesos aborda las prácticas de los proyectos de desarrollo y mantenimiento de software. Esta categoría realiza las actividades de acuerdo a los elementos proporcionados por la Categoría de Gerencia y entrega a ésta la información y productos generados.

2.4.3 Procesos de Moprosoft

Como puede notarse en la figura 2.4, cada categoría de MoproSoft está formada por uno o varios procesos, cuyos propósitos son (Oktaba, Alquicira, Su, & Martínez, 2005, pág. 14):

- Para la Gestión de Negocio es establecer la razón de ser de la organización, sus objetivos y las condiciones para lograrlos, considerando las necesidades de los clientes, así como evaluar los resultados y proponer cambios que permitan la mejora continua. Adicionalmente la gestión de negocio habilita a la organización para

responder a un ambiente de cambio y a sus miembros para trabajar en función de los objetivos establecidos.

- El propósito de Gestión de Procesos es establecer los procesos de la organización, en función de los requeridos e identificados en el plan estratégico. También se encarga de definir, planificar, e implantar las actividades de mejora en los mismos.
- El propósito de la Gestión de Proyectos es asegurar que los proyectos contribuyan al cumplimiento de los objetivos y estrategias de la organización.
- El propósito de Gestión de Recursos es conseguir y dotar a la organización de los recursos humanos, infraestructura, ambiente de trabajo y proveedores, así como crear y mantener la base de conocimiento de la organización. Igual que los otros procesos, este busca apoyar el cumplimiento de los objetivos del plan estratégico de la organización.
- El propósito de Recursos Humanos y Ambiente de Trabajo es proporcionar el capital humano adecuado para cumplir las responsabilidades asignadas a los roles dentro de la organización, así como la evaluación del ambiente de trabajo.
- El propósito de Bienes, Servicios e Infraestructura es proporcionar proveedores de bienes, servicios e infraestructura que satisfagan los requisitos de adquisición de los procesos y proyectos.

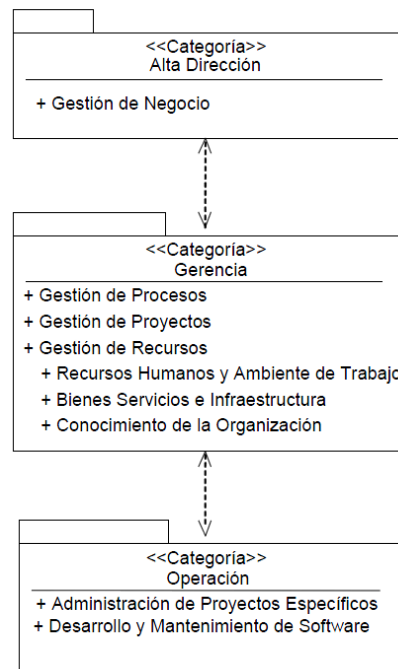


Fig. 2-5 Diagrama de categorías de procesos. (Oktaba, Alquicira, Su, & Martínez, 2005).

- El propósito de *Conocimiento de la Organización* es mantener y administrar la base de conocimiento que contiene la información y los productos generados por la organización.
- El propósito de la *Administración de Proyectos Específicos* es establecer y llevar a cabo sistemáticamente las actividades que permitan cumplir con los objetivos de un proyecto en tiempo y costo esperados.
- El propósito de *Desarrollo y mantenimiento de Software* es la realización sistemática de las actividades de análisis, diseño, construcción, integración y pruebas de productos de software nuevo o modificado cumpliendo con los requerimientos especificados.

2.5 Estado del arte

Existe una gran cantidad de información relacionada a los temas que abarca esta investigación. Los modelos de mejora tienen su origen en el *Software Engineering Institute* (SEI) de *Carnegie Mellon University*, en Estados Unidos (Watss H., 1988). Se iniciaron a través de un marco conceptual y más tarde a través

del modelo de madurez de los procesos de software (Mark P., Bill C. & Beth C., 1993). Algunos de los trabajos más relevantes relacionados al modelo inicial propuesto en (Carnegie, 1995), así como relacionados a capacitación y a diseño ágil se describen a continuación.

El trabajo propuesto por (Visconti M. & Cook C., 1998) propone niveles de madurez para la verificación de los documentos de requerimientos, diseño y pruebas como una manera asegurar la calidad en el desarrollo El *E-Learning Maturity Model* (Marsall S. & Mitchel G., 2006) ofrece un medio por el que las instituciones pueden evaluar y comparar su capacidad para desarrollar, implementar y dar soporte a plataformas de *e-learning*. El trabajo expuesto por (Lutteroth C., Luxton A, Dobbie G. & Hamer J., 2007) se puede utilizar para calificar las organizaciones educativas, de acuerdo a su capacidad para ofrecer una calidad educativa en una escala de cinco niveles. Por otra parte, este mismo modelo se puede utilizar para mejorar la capacidad de una institución mediante la aplicación de las mejores prácticas y los cambios organizativos que describe. Otra propuesta está dada por (Kadjo M., Forssander S. & Olsson U., 2001) donde, mediante un conjunto de niveles y procesos, propone la enseñanza y entrenamiento de la fase de mantenimiento en el desarrollo del software. También en la fase de pruebas de software se ha desarrollado por parte de la *TMMi* (Veenendaal, 2010) un modelo de madurez llamado *Testing Maturity Model*, que mediante la misma filosofía proporciona mejoras a la actividad de pruebas de software. Otro acercamiento en el campo de la capacitación en el área de la Ingeniería de Software es el presentado por Fraga y colaboradores (Fraga A. & Llorens J., 2007) quienes proponen un modelo de mejora de procesos basado en el uso de niveles, para clasificar a las instituciones que imparten la materia de arquitecturas de software. Este trabajo es un esquema de entrenamiento para la enseñanza, que se clasifica en 5 niveles de conocimiento y entrenamiento. Otro trabajo interesante es el de Klappholdz y colaboradores (Klappholz, Bernstein, & Port D, 2003), quienes proponen una herramienta que mide el rendimiento de los estudiantes en la implementación de la mejora de procesos. De los trabajos revisados no se han hallado

a la fecha esfuerzos para la construcción de un modelo de mejora para la capacitación en empresas de desarrollo de software.

2.5.1 Algunos modelos de capacitación de desarrollo de software

En el trabajo propuesto por (Chuan-Hoo & Week-Kek, 2010), se enfoca a la capacitación para desarrollar los conceptos de agilidad, flexibilidad y adaptabilidad (Reichlmayr, 2003). El curso en sí exhibe 5 componentes distintivos, el primero consiste en la incorporación de metodologías ágiles híbridas con las mejores prácticas de codificación y administración de proyectos. El segundo componente es una introducción al conflicto de los requerimientos cambiantes en los negocios. El tercero consiste en una guía constante a los equipos de estudiantes actuando los catedráticos como usuarios del negocio. El cuarto es el énfasis constante en la creación de pequeños equipos de trabajo, entre 4 a 6 integrantes. Cabe hacer notar que el tamaño de los equipos es una característica propia de los equipos ágiles, ya que con esto se consigue una mejor comunicación para enfrentar los cambios de requerimientos. El quinto componente de este curso consiste en dar un énfasis en el desarrollo incremental. La evaluación de dicha investigación se realizó mediante análisis estadísticos de los datos recabados durante la aplicación del experimento. Éste se llevó a cabo durante 4 semestres haciendo uso de 82 equipos, encontrándose una mejora significativa en el uso y manejo de diversas características del desarrollo ágil.

Otro de los trabajos de investigación es el expuesto por (Wingreen & LeRouge, 2009). Aquí se busca una clasificación de las preferencias de capacitación de los profesionistas en Tecnologías de la Información (TI) y determinar las preferencias que están relacionadas a las mismas. Para esto, los autores hacen uso de una metodología Q. La metodología Q es una técnica para medir objetos o afirmaciones, es decir, es un método de ordenamiento por rango de actitudes o juicios. Este método resulta efectivo cuando el número de elementos es gran magnitud (Stephenson, 1953). Utilizando la metodología Q, Wingreen y LeRouge identificaron algunos de los trabajos de capacitación que tienen mayor aceptación dependiendo de la carrera que el entrevistado tiene. El estudio permite a los

administradores de TI tener información para poder construir sus cursos de capacitación dependiendo a los recursos interpretativos del modelo propuesto.

Dentro de los esfuerzos de capacitación de metodologías ágiles a nivel academia, se encuentra el propuesto por (Alfonso & Botío, 2005). Este trabajo desarrolla un modelo ágil de procesos iterativos en un curso de ingeniería de software a nivel posgrado. Esta técnica sirve tanto para la capacitación de los alumnos como de académicos; la forma de evaluar este esfuerzo es la comparación de recientes experiencias de desarrollo con los resultados obtenidos mediante el uso del modelo, es decir, hubo en base a un historial académica de 3 años anteriores, existiendo mejoras significativas en los tiempos de entrega y trabajo realizado. Por ejemplo, el tiempo de entrega del trabajo de los alumnos se redujo en un 50% esto debido a la entrega ágil aprendida, por otra parte, el trabajo de los profesores también se vio beneficiado en el registro de las notas de los estudiantes ya que la entrega en varias iteraciones permite ir revisando y teniendo un mejor control. Dentro de los casos de éxito de la implementación de equipos ágiles mediante una capacitación previa está la expuesta por (Prochazka, Kokott, Chmelar, & Krchnak, 2011). En este trabajo, mediante un curso ágil distribuido entre equipos holandeses e hindúes se realizó una capacitación para el desarrollo de proyectos adoptando técnicas ágiles. Las fases que se manejaron en este trabajo fueron cuatro: mapeo, activación, tutoría e independencia. Mediante dos casos de estudio se dirigió la capacitación, teniendo en cuenta la estimación ágil, los niveles de planeación, el trazado del progreso, definición de hecho, la definición de calidad por parte de los clientes, retroalimentación del cliente, la cooperación y el trabajo del equipo mediante la estimación y planes. Estos puntos se tomaron de una pregunta clave que se les realizó a los integrantes de los equipos ¿Por qué quieres mejorar?; el esfuerzo que se obtuvo al finalizar el curso se ve reflejado por parte del equipo y de la satisfacción del cliente. Los equipos compararon resultados obtenidos de anteriores proyectos con respecto a las variables de planeación y desempeño, consiguiendo una disminución; por parte del cliente se hicieron uso de encuestas de satisfacción, las cuales fueron positivas en relación a los tiempos de entrega.

Capítulo 3 METODOLOGÍA

Para llevar a cabo este proyecto, se inició con un análisis del uso de las metodologías ágiles en la industria mexicana desarrolladora de software. Esto se hizo con el fin de verificar el grado de pertinencia de este tipo de metodologías en las empresas verificadas en el nivel uno de la norma NMX-I-059-NYCE-2005. Asimismo, se especificaron las propiedades y rasgos importantes asociados al desarrollo ágil que se implementaron en las PyMEs que participaron en este proyecto. Con esto es posible evaluar los beneficios que resultarían de la implementación del modelo propuesto en el presente trabajo.

3.1 Alcance y limitaciones

La ingeniería de software incluye modelos, estándares y métodos altamente probados experimentalmente y que el/la profesionista podría aplicar, pero resulta muy frustrante cuando los recién graduados y/o desarrolladores con experiencia no consiguen adaptarlos a las necesidades de la empresa de desarrollo y en consecuencia se dificulta enormemente su inserción en el ámbito laboral. Por otra parte, sabemos que en la actualidad es solicitado el uso de estándares y modelos de desarrollo por muchas compañías, tanto a nivel nacional como internacional. Sin embargo en la mayoría de los casos, para poder conocerlos y aplicarlos, las empresas pequeñas deben erogar en gastos de capacitación, pues muchas veces este conocimiento no está incluido dentro de los planes y estudios universitarios, dando pauta a que sus egresados carezcan de disciplina en el uso de los mismos. El alcance del presente trabajo implica que las PyMEs adopten un modelo de mejora del proceso de capacitación, para que los integrantes del desarrollo se disciplinen en ambientes de trabajo con filosofías de desarrollo ágil. Con esto puede conseguirse una adquisición de competencias que permitan mejorar la producción, mantener los proyectos dentro de un presupuesto y tiempo de desarrollo establecidos.

La limitación del modelo propuesto es el de trabajar en los nivel cero y uno de la norma NMX-I-059-NYCE-2005. Con esto el contexto de validación y prueba del modelo está acotado a este tipo de empresas.

3.2 Hipótesis

Nuestra hipótesis supone la existencia de las siguientes premisas en el ambiente donde se implementaría el modelo propuesto CAAM:

- el desarrollo de software en la actualidad debe ser ágil y producir entregables frecuentes para satisfacer las necesidades del cliente (VisionOne, 2010);
- una parte importante de la población profesional dedicada al desarrollo de software cuenta con buenas habilidades de análisis, síntesis y programación, pero les dificulta el esquema de planeación ágil (Olivos, 2010);
- la mayoría de los desarrolladores mexicanos realizan pruebas manuales y post-código (Olivos, 2010);
- la mayoría programadores mexicanos no realizan ningún tipo de medición de sus procesos (Rodriguez, 2006);
- aún y cuando Moprosoft es considerado un umbral para saltar al cumplimiento de modelos internacionales de calidad, la mayoría de las empresas verificadas con Moprosoft 1, en la práctica se les hace complejo (Nava, 2009);
- una importante cantidad de desarrolladores no pueden dedicar más de 40 horas al año en capacitaciones para actualizarse.

La hipótesis de la investigación es la siguiente:

“Es posible definir un modelo de mejora del proceso de capacitación de empresas PyMEs verificadas en el nivel 1 de Moprosoft, que permita mejorar las competencias de sus desarrolladores, considerando que cumplen las premisas anteriores y que utilizan modelos ágiles”

3.3 Diseño de la investigación

Según Hernández y colaboradores, “El término diseño se refiere al plan o estrategia concebida para obtener información que se desea. El diseño señala al investigador lo que debe hacer para alcanzar sus objetivos de estudio y para contestar las interrogantes de conocimiento que se ha planteado” (Hernández, Fernández, & Baptista, 2007, pág. 159). El enfoque de investigación utilizado en este trabajo fue de tipo cuantitativo, ya que se recolectaron datos para una medición numérica y análisis estadístico, estableciendo patrones de comportamiento. Esto concuerda con la definición dada por Hernández: “Enfoque cuantitativo: usa recolección de datos para probar hipótesis con base en la medición numérica y el análisis estadístico para establecer patrones de comportamiento” (Hernández, Fernández, & Baptista, 2007, pág. 6). El tipo de diseño de la investigación es de “pre-prueba, post-prueba” con un sólo grupo. A un grupo se le aplica una prueba previa al estímulo o tratamiento experimental, después se le administra un curso diseñado utilizando el modelo CAAM y finalmente se le aplica una prueba posterior al tratamiento. Se realiza la recolección de los datos con la técnica de aplicación de cuestionarios (inicial y final) a las empresas desarrolladoras de software que cumplan con las premisas iniciales descritas en la sección 3.2.

3.4 Selección de la muestra

La población considerada en esta investigación está formada de pequeñas y medianas empresas⁴ desarrolladoras de software, que cuenten con el nivel cero o uno de la norma mexicana NMX-I-059-NYCE-2005 (Moprosoft). El tipo de muestreo empleado es el no probabilístico, por lo que se consideró a un grupo de la población definido por las características consideradas en el presente trabajo. Este muestreo va de acuerdo a la definición dada por Hernández con respecto a una muestra no probabilística: “subgrupo de la población en la que la elección de los elementos no

⁴ En el contexto de esta investigación, se consideran empresas pequeñas y medianas a aquellas que tienen entre 3 y 25 empleados

depende de la probabilidad sino de las características de la investigación” (Hernández, Fernández, & Baptista, 2007, pág. 262).

3.5 Recolección de datos

Dentro de esta fase del trabajo, inicialmente se llevó a cabo una encuesta que se difundió vía web durante aproximadamente 3 meses, desde el portal de la Universidad Politécnica de Tulancingo⁵ (ver encuesta en el Anexo A). El objetivo de este cuestionario fue recabar datos con respecto al comportamiento de las empresas de nivel cero o uno de Moprosoft que estuvieran haciendo uso de las metodologías ágiles. Del total de 50 registro obtenidos se seleccionó una muestra representativa de 38 cuestionarios que se contestaron desde diversos estados de la República Mexicana; los restantes eran incorrectos o estaban sin contestar. Cabe hacer notar que un 45% de las empresas encuestadas poseían el nivel 1 de Moprosoft, y el 55% restante contaban con un nivel diferente de uno (cero).

La figura 3.1 muestra la experiencia de los encuestados en la aplicación de metodologías ágiles en estas empresas. Como se puede observar, el uso de las metodologías ágiles puede considerarse una oportunidad para incorporarse en empresas con nivel cero o uno de Moprosoft. El porcentaje más alto obtenido correspondió a empresas donde los integrantes tienen un desconocimiento del uso de tales técnicas y solamente 15 empresas de las encuestadas dijeron tener un buen tiempo usando y obteniendo beneficios de tales técnicas.

Otra de las secciones del cuestionario se dedicó a identificar el nivel de experiencia y adopción de las metodologías ágiles en los miembros de la organización. Se definieron los siguientes niveles: 1) Adopción a nivel organizacional; 2) Adopción a nivel de equipo; 3) Adopción a nivel personal. Los resultados se muestran en la figura 3-2.

⁵ Estuvo disponible en la dirección <http://www.upt.edu.mx/cuestionario.php>



Fig. 3-1 Grado de adopción de las metodologías obtenido en 38 empresas encuestadas.

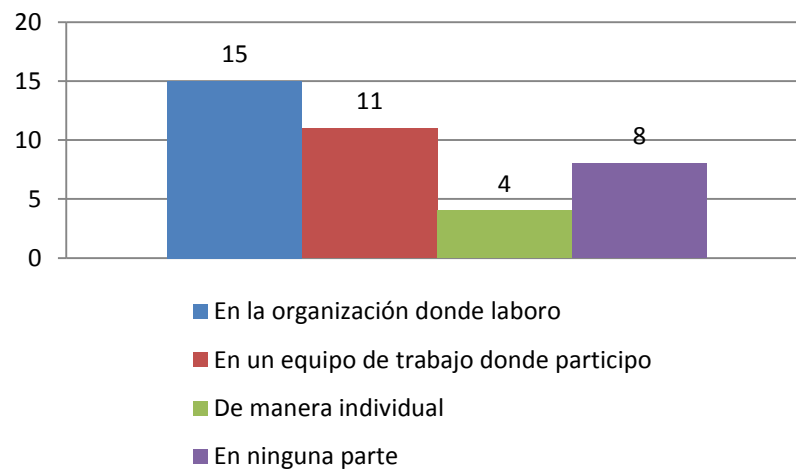


Fig. 3-2 Nivel de adopción de metodologías ágiles obtenido en 38 empresas encuestadas.

La agilidad requiere de entregas constantes y generación de productos de valor para el usuario. La capacidad de adaptación de los desarrolladores a estas circunstancias marca una diferencia en las organizaciones que son exitosas en el mercado del software. La figura 3-3 muestra los factores más importantes que motivan la adopción de las metodologías ágiles en las empresas encuestadas.

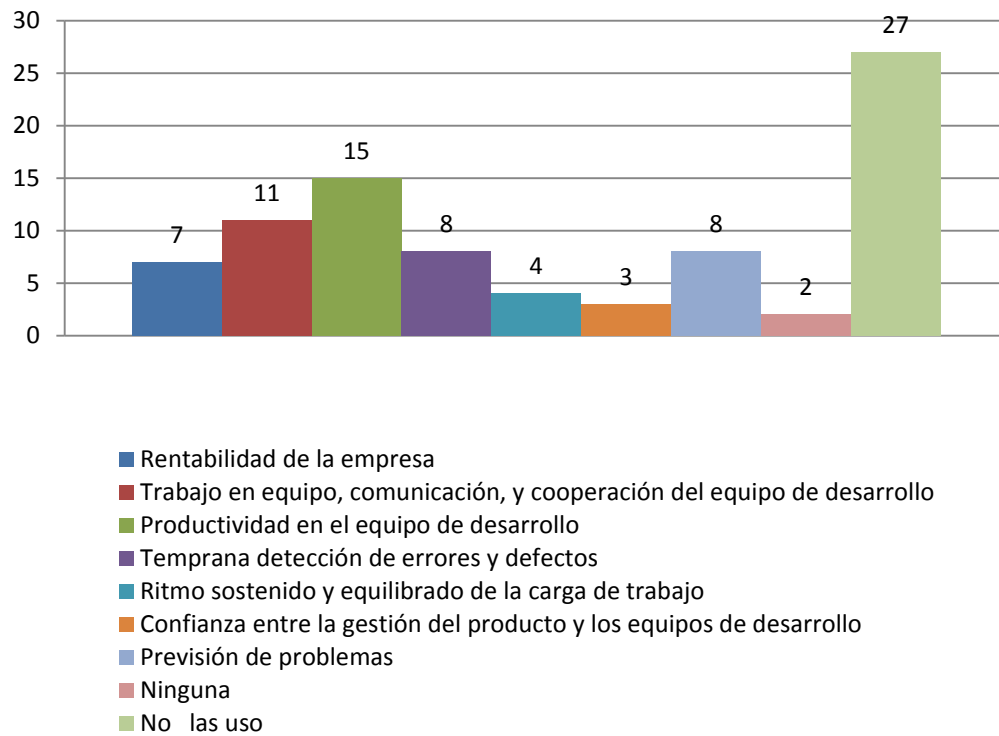


Fig. 3-3 Motivos de adopción de las metodologías ágiles.

Como siguiente paso relacionado a la recolección de datos, se seleccionaron empresas⁶, en donde se implementó CAAM. Los desarrolladores de estas empresas involucrados en el experimento contestaron el cuestionario descrito en el anexo E, antes de realizarse el experimento de implementación, y el cuestionario descrito en el anexo F después de realizarse el experimento. Dichos cuestionarios en conjunto contienen 33 reactivos, proporcionaron información relacionada a:

- Planeación de actividades para ambientes ágiles.
- Comunicación con el cliente del producto
- Uso de herramientas ágiles.
- Uso de herramientas de pruebas.

Durante el diseño de estos cuestionarios se buscó que fueran válidos y confiables, siguiendo la recomendación dada por Hernández: “Toda medición o instrumento de recolección de datos debe de reunir dos requisitos esenciales: confiabilidad y validez,

⁶ Estas empresas son diferentes a las que contestaron el primer cuestionario

cabe agregar que un instrumento de medición puede ser confiable pero no necesariamente válido, por ello es requisito que el instrumento de medición demuestre ser confiable y válido” (Hernández, Fernández, & Baptista, 2007, pág. 288).

Los cuestionarios se diseñaron siguiendo el método de escalamiento de Likert (Hernández, Fernández, & Baptista, 2007, pág. 341), tomando en cuenta las variables involucradas en la hipótesis. Se aplicó el modelo a empresas para la obtención de datos que posteriormente se trataron mediante técnicas estadística descriptiva para obtener el comportamiento.

3.6 Aplicación del instrumento

La metodología de la investigación usada en este trabajo es mediante el manejo de encuestas para la administración de la información. En términos generales, consiste de cuatro fases:

Fase I: Formulación de las pregunta de investigación, construcción de un conjunto de premisas e hipótesis.

Fase II: Diseño e implementación del modelo CAAM.

Fase III: Diseño del instrumento de encuesta y la recolección de los datos mediante el mismo.

Fase IV. Desarrollo de taller de Capacitación en Scrum-Moprosoft, el que se llevo a cabo en la ciudad de Puebla e Hidalgo contando con una participación de 26 participantes de empresas tales como: Synergy, Zutri, BTOB Consultores Eviciti. El mecanismo que se llevo a cabo en el taller fue bajo los siguientes puntos:

- Inducción a las metodologías ágiles.
- Facilitación de conceptos ágiles en la práctica
- Fase de independencia para la utilización de *Scrum* en algún proyecto de la empresa.
- Mapeo de actividades propias de Moprosoft en *Scrum*

Fase V: Análisis de los datos obtenidos en la fase III.

Capítulo 4 EL MODELO PROPUESTO

En este capítulo se presenta la descripción del modelo de capacitación propuesto, dando a conocer los elementos que conforman su diseño. Asimismo, se realiza un comparativo entre el modelo de mejora de proceso a nivel nacional Moprosoft y la gestión de proyectos (*Scrum*), con el objetivo de proporcionar elementos para el desarrollo de un modelo de capacitación basado en ambientes ágiles. El rápido ritmo de cambio de la tecnología del software significa que sus profesionales se enfrentan a una continua necesidad de aprender nuevas habilidades, por lo que requieren usar mecanismos fáciles de adoptar y que aseguren el desarrollo de sus competencias.

4.1 El Modelo CAAM

En este trabajo se propone un modelo llamado “Modelo de Capacitación de Ambientes Ágiles bajo Moprosoft” (CAAM), dirigido a las pequeñas y medianas empresas desarrolladoras de software. CAAM considera un proceso para la detección de necesidades, planeación y ejecución de la capacitación, teniendo como meta el desarrollo de las competencias necesarias para el dominio del desarrollo de software usando técnicas ágiles. CAAM da como resultado valores que a la organización le son de utilidad, que se espera se vean reflejados en la competitividad y el esfuerzo del desarrollo de software por parte de los integrantes del equipo.

CAAM considera la conjunción de tres filosofías, según se muestra de manera general en la figura 4-1. Contando con una técnica ágil para la gestión de proyectos que es *Scrum* como un elemento para la dirección del trabajo en equipo que es aplicado en este caso para el desarrollo de software, se usa a su vez el modelo de mejora de procesos de moprosoft (nivel 1) como elemento rector del cumplimiento de las principales actividades que se deben cumplir para la producción de software, además de tener como elemento de la construcción del software a TDD. Elementos vistos en el capítulo 2. CAAM está integrado por un área clave, que a su vez contiene actividades para el logro de la implementación de las competencias. Para desarrollar

el modelo adecuadamente, se desarrolló un proceso de integración Moprosoft-Scrum. Moprosoft es un modelo de procesos de desarrollo y mantenimiento de software a nivel nacional⁷, que en su categoría de operación incluye los procesos “Administración de procesos específicos” y “Desarrollo y Mantenimiento de Software”. Por otra parte Scrum es una metodología ágil, donde su modo de trabajo se centra en la producción incremental, estableciendo períodos cortos de trabajo en los que se sigue un mismo patrón.

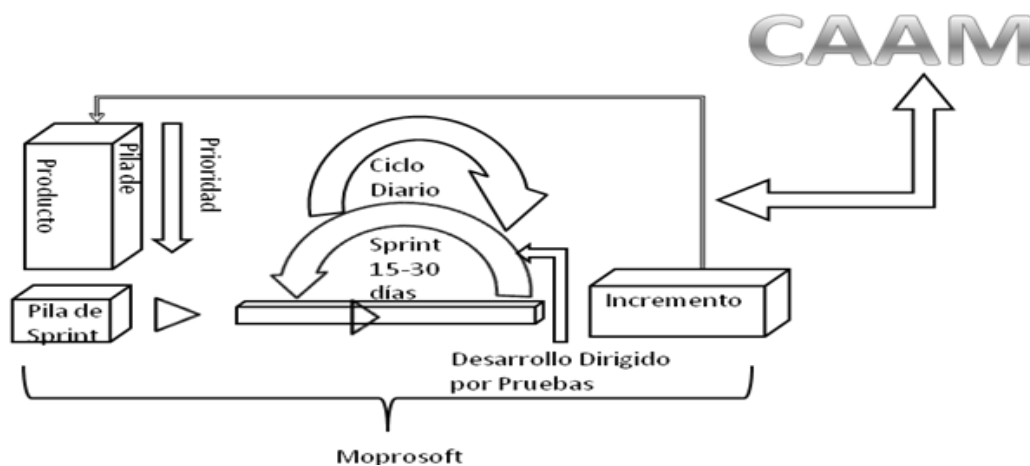


Fig. 4-1 CAAM y las filosofías de trabajo Moprosoft, Scrum y desarrollo TDD.

CAAM, como todo modelo de mejora de procesos, se forma por un conjunto de niveles que, mediante el cumplimiento de su práctica clave, permiten una evolución. CAAM se aplica a las empresas que estén en el nivel 1 de Moprosoft. La capacitación en metodologías ágiles aplicadas a los procesos de Moprosoft “Administración de proyectos específicos; y Desarrollo y mantenimiento de Software,” permite dominar el uso de herramientas y la aplicación de conocimientos que llevarán a desarrollar la competencia en el concepto de agilidad en la aplicación de Moprosoft.

4.2 Competencias a desarrollar

La competencia es un elemento que incide en el dominio de los conocimientos y habilidades que se deben adquirir de manera individual. En el modelo propuesto

⁵Se define en Capítulo 2 sección “Moprosoft”

CAAM se hace uso del desarrollo de las competencias como parte del proceso clave y de esta manera cumplir con la competitividad de los equipos de trabajo. Cabe hacer notar que, para definir el enunciado de una competencia, se hace uso de la estructura gramatical: “*Verbo + Objeto + Complemento + Condición*” según lo propuesto en (Catalano A. M.; Avolio de Cols S.; Sladogna M., 2004). En las tablas D.1 y D.2 del anexo D se presentan las competencias que se deben de cubrir tanto en CAAM, para tener un seguimiento de los conocimientos y habilidades que los desarrolladores deben de adquirir (nivel 1 de Moprosoft), para la categoría de operación.

4.3 Estrategia de conformación del CAAM

Para diseñar a CAAM se consideraron algunos elementos propuestos por Buckley (Buckley R. Caple J., 2009)(Buckley & Caple, 2009) asociados al entrenamiento⁸. Por otra parte se realizó una comparación entre las actividades en común o parecidas de las filosofías Moprosoft y *Scrum*; el detalle de esta comparación se muestra en los anexos B y C. Este ejercicio de comparación sirvió de base para determinar el proceso de capacitación que se tiene que llevar a cabo para la adquisición de las competencias en el modelo de CAAM. Moprosoft se caracteriza por estar constituido por procesos y actividades que presentan cierta rigidez en sus actividades. La segunda filosofía, *Scrum*, es un método de gestión ágil, caracterizado entre otras cosas por entregas constantes y coraje para aceptar los cambios, con la finalidad de que los desarrolladores adquieran características propias de la filosofía ágil se adecua una herramienta de entrenamiento.

4.4 Relación de actividades entre *Scrum* y Moprosoft

Las dos primeras fases de *Scrum*⁹, conocidas como “Pila de producto” y “Reunión de planificación de *Sprint*” tienen elementos comunes con Moprosoft en el

⁸ Ver capítulo 2 para una definición detallada del modelo de Buckley y Caple.

⁹ Se explica de manera de detalle en el Capítulo 2, sección “*Scrum*”.

proceso de “Administración de proyectos específicos”. Las fases “Revisión” y “Retrospectiva” tienen puntos en común con el proceso de “Desarrollo y Mantenimiento de Software” de Moprosoft. La figura 4-2 muestra esta relación de procesos entre Moprosoft, Scrum y TDD. En esta figura, los círculos corresponden a fases de *Scrum*, y la línea punteada divide las áreas que corresponden a los procesos de Moprosoft “Administración de Proyectos específicos” y “Desarrollo y Mantenimiento.” Enseguida se explica con detalle la relación de cada componente de *Scrum* con Moprosoft. Para una explicación detallada de cada componente, ver la sección 2.9 y 2.10.

1) Pila de producto. Contiene la lista de objetivos/requisitos priorizados del proyecto, y representa la visión y expectativas que tiene el/la cliente del producto o proyecto. Esta fase se relaciona con las siguientes actividades de Moprosoft correspondientes al proceso de “Administración de procesos específicos:” A1.9, A1.10, A1.13 y A1.22; para el proceso de “Desarrollo y mantenimiento de Software” las actividades que se tienen que realizar en esta fase son A2.2 y A2.5.

2) *Reunión de planeación de la iteración*. Es una reunión entre el dueño del producto y el equipo. Llevándose a cabo en cada uno de los *sprints*, para definir que se hará en las siguientes iteraciones. Esta fase se relaciona con las siguientes actividades de Moprosoft del proceso de “Administración de procesos específicos:”

- A1.3 Definir conjuntamente con el cliente el Protocolo de entrega de cada uno de los entregables especificados en la “Descripción del proyecto”,
- A1.4 Identificar el número de ciclos y las actividades específicas que deben llevarse a cabo para producir los entregables y sus componentes identificados en la Descripción del Proyecto,

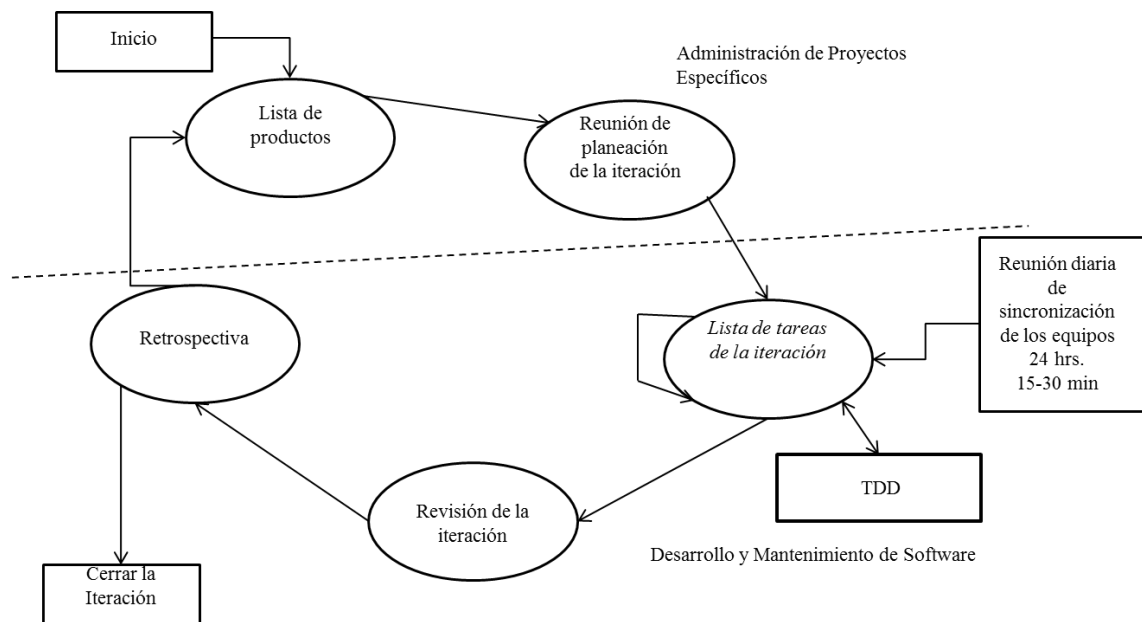


Fig. 4-2 División de procesos de Scrum a Moprosoft

- A1.6 Determinar el tiempo estimado para cada actividad, considerando las metas cuantitativas para el proyecto,
- A1.8 Establecer el equipo de trabajo que realizará el proyecto, establecer el calendario de actividades,
- A1.10 Calcular el costo estimado del proyecto,
- A1.11 Definir el plan de riesgos.

3) Lista de tareas de iteración. Es una lista de tareas que el equipo elabora en la reunión de planificación de la iteración (Planeación de la iteración) como plan para completar los objetivos y requisitos seleccionados para la misma. Esta fase se relaciona con las siguientes actividades de Moprosoft del proceso de “Desarrollo y mantenimiento de Software”, en el que prácticamente es la realización sistemática de las actividades de análisis, diseño construcción, integración y pruebas de producto de software:

A1.1. Revisar con los miembros del equipo de trabajo el Plan de Desarrollo actual para lograr un entendimiento común y obtener su compromiso con el proyecto.

A2.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al Plan de Desarrollo actual.

A2.2. Analizar los requerimientos identificados para delimitar el alcance y su factibilidad, considerando las restricciones del ambiente del negocio del cliente o del proyecto.

A3.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al Plan de Desarrollo actual.

A3.2. Documentar o modificar el Análisis y Diseño:

- Analizar la Especificación de Requerimientos para generar la descripción de la estructura interna del sistema y su descomposición en subsistemas, y éstos a su vez en componentes, definiendo las interfaces entre ellos.

A4.2. Construir o modificar el (los) componente(s) de software:

- Implementar o modificar Componente(s) con base a la parte detallada del Análisis y Diseño.

A4.5. Incorporar componentes a la configuración del software.

A5.2. Realizar integración.

- Integrar los componentes en subsistemas o en el desarrollo de Software.

Debido a que en las metodologías ágiles se lleva a cabo poca actividad de documentación, las actividades A5.3, A5.8, A5.11 de Moprosoft (ver anexo B) no se mapean en la metodología ágil, por lo que se deberá de realizar conjuntamente en el “Pila de iteración” la documentación que se marca en Moprosoft, de manera que se cumpla con dichas actividades.

- 4) *Retrospectiva*. Con el objetivo de mejorar de manera continua su productividad y la calidad del producto que está desarrollando, el equipo analiza cómo ha sido su manera de trabajar durante la iteración y el porqué está consiguiendo o no los objetivos a los que se comprometió al inicio de la iteración.

5) *Reunión diaria*. El objetivo de esta reunión es facilitar la transferencia de información y la colaboración entre los miembros del equipo para aumentar su productividad, al poner de manifiesto puntos de colaboración.

4.5 Relación de roles entre Scrum y Moprosoft

En la literatura de Programación Extrema (XP, por sus siglas en inglés) se considera generalmente que un equipo ágil debe contar con los siguientes miembros: diseñador(a), administrador(a) de proyectos, desarrolladores, analistas y probadores (Rassmusson J., 2010). En el caso de *Scrum* se añade a esta lista dos roles; el facilitador (*Scrum Master*) y el dueño del producto (*Product Owner*) (Schwaber & Beedle, 2001). Por otra parte, Moprosoft cuenta con los siguientes roles en su proceso de “Administración de Procesos Específicos”: responsable de gestión de proyectos, responsable de la administración del proyecto específico, cliente, responsable del subcontrato, responsable de desarrollo y mantenimiento de software. Moprosoft requiere de los siguientes roles: responsable de desarrollo y mantenimiento de software, analista, diseñador(a) de interfaz de usuario, diseñador(a), programador(a), responsable de manuales y usuarios (Oktaba, Alquicira, Su, & Martínez, 2005). La Tabla B.1 del anexo B, presenta una comparación entre los roles indispensables para el nivel 1 de Moprosoft correspondientes a la categoría “Administración de Proyectos Específicos” y Scrum.

Por otra parte, es necesario determinar la relación entre las actividades de Scrum y Moprosoft; la tabla B.2 del anexo B, muestra una comparación entre éstas. En la categoría de “Administración de Proyectos Específicos” fase de Planificación se identificaron 12 tareas, las cuales se detallan en el mismo anexo. De manera similar, en el anexo C se comparan los roles definidos como indispensables para el nivel 1 de Moprosoft para el proceso “Desarrollo y Mantenimiento de Software” correspondiente a la categoría de “Operación.”

4.6 Niveles en el modelo de CAAM

La arquitectura general de CAAM se muestra en la figura 4-3. El modelo se encuentra formado por 4 niveles de capacitación: inicial, repetible, definida y administrada. Cada uno de estos niveles tiene procesos a cumplir, permitiendo la evolución al siguiente nivel. Enseguida se describe cada nivel.

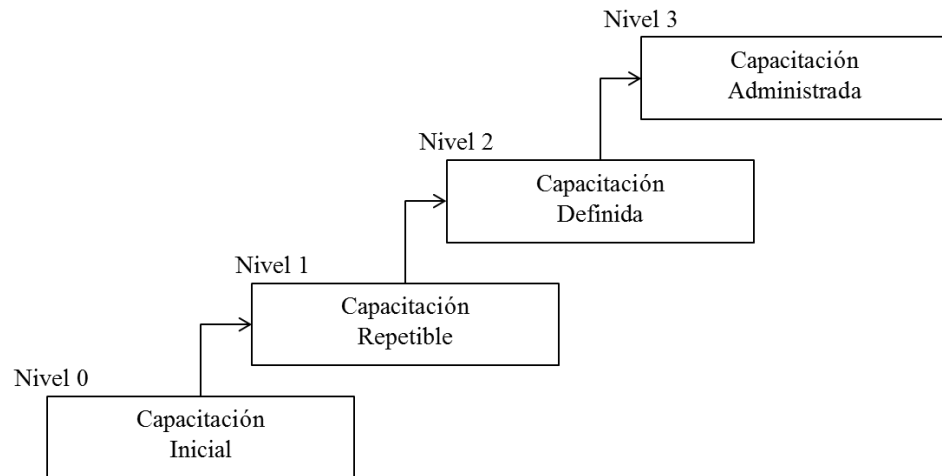


Fig. 4-3 CAMM: Modelo de madurez para la capacitación.

Nivel 0. La empresa u organización no cuenta con ningún proceso formal de capacitación para cualquier actividad inherente al desarrollo de software, o se tienen sin conocer el “por qué,” “para qué” y/o los costos que implica ésta. Además, no existe documentación de la especificación de estos procesos. En muchos de los casos la capacitación se lleva a cabo de manera “caótica.”

Nivel 1. Se establece un mecanismo de planificación en base a la demanda de capacitación individual, en equipo o de forma organizacional. Se identifican las competencias a lograr como hito de las etapas del desarrollo; el conocimiento de cada una de ellas es de importancia para el logro de los siguientes niveles. Se tienen algunos procesos no formalizados, además de contar con conocimientos previos de capacitaciones anteriores las cuales se toman de base para alguna toma de decisiones.

Nivel 2. Se realiza el proceso de capacitación para los hitos identificados a nivel organización, obteniendo métricas de los componentes del proceso de capacitación, básicamente de producto y de organización. Una vez terminada la capacitación, se

debe de asegurar la competencia en la implementación de técnicas y artefactos ágiles. Se hace uso de los procesos ya establecidos por parte de la organización, además se realiza la medición de los mismos para corroborar la calidad del proceso y producto. La capacitación pasa a ser un proceso fundamental para el logro de los objetivos estratégicos.

Nivel 3. Este nivel sirve de base para determinar la calidad del proceso de capacitación, utilizando los resultados cuantitativos a nivel individual, de equipo y a nivel organización. Los resultados obtenidos son elementos claves para facilitar la retroalimentación del proceso de capacitación, además de las propuestas para la mejora de los procesos actuales, en caso de necesitarse. Se establecen mecanismos de capacitación basándose en herramientas de tecnologías de la información.

4.7 Área clave de proceso para CAAM

Normalmente cada nivel de madurez de un modelo de mejora está compuesto por varias áreas claves de procesos, las que a su vez contiene metas y actividades de tipos genéricas y específicas. La figura 4-4 muestra la estructura de la única área clave de proceso utilizada para la implantación de CAAM. El propósito de esta área de proceso es desarrollar la capacitación mediante un conjunto de prácticas específicas, las cuales a su vez, contienen un conjunto de actividades para el logro de las competencias asociadas al modelo de capacitación.

4.7.1 Prácticas específicas de CAAM

CAAM contiene prácticas que, mediante actividades, guían de manera detallada las acciones que se realizan para la implementación de las competencias. Cada práctica a su vez contiene un conjunto de actividades para lograr su finalidad. En los siguientes párrafos se describen cada una de las prácticas. Las actividades se identifican a través de un número, para facilitar el proceso de evaluación del CAAM.

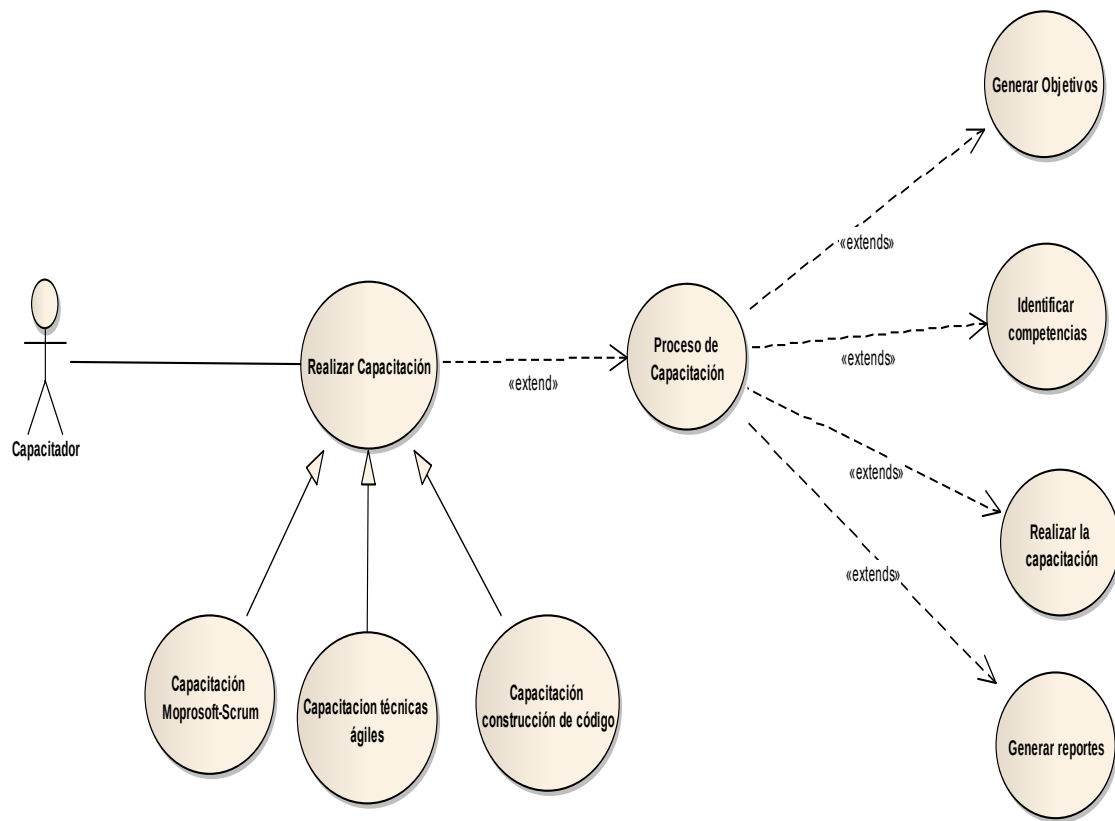


Fig. 4-4 Área clave de proceso de CAAM.

Práctica específica 1: Generar Objetivos

A1.1 Establecer un objetivo de capacitación y políticas de dirección, dependiendo de las necesidades organizacionales.

A1.2 Identificar las habilidades y conocimientos que se esperan obtener según el objetivo de la capacitación, mediante el método de DACUM.

A1.3 Priorizar los cursos de capacitación dependiendo de la jerarquía y peso del mismo para la organización en consenso con los miembros del equipo.

A1.4 Conformar una matriz de competencias donde se identifiquen las unidades de aprendizaje, resultados de aprendizaje, evidencias y técnicas de evaluación.

A1.5 Establecer los mecanismos de la capacitación, estrategias de transmisión del conocimiento y el desarrollo de las habilidades en las que se pueden contar a la instrucción tradicional o seminarios por la web.

A1.6 Establecer los recursos necesarios como son las herramientas, ambientes y el personal que impartirá la capacitación.

A1.7 A partir de una situación hipotética realizar una aplicación haciendo uso de los

elementos a capacitar.

Práctica específica 2: Identificar Competencias

A2.1 Desarrollar la matriz de competencia para cada área estratégica de enseñanza (Moprosoft-*Scrum*, herramientas ágiles y construcción de código).

A2.2 Validar y verificar las competencias con el equipo y dueño del producto.

A2.3 Desarrollar la competencia en la especificación de requerimientos, para el cumplimiento de las necesidades y expectativas acordadas incluyendo las pruebas de usabilidad de la interfaz de usuario.

A2.4 Desarrollar la competencia en la conformación de una pila de producto, determinar la importancia y establecer estimaciones en las historias de usuario, para la Documentación o modificación de la Especificación de Requerimientos.

A2.5 Desarrollar la Competencia en el uso de las principales herramientas para la administración de un *Sprint* en el nivel 1 de Moprosoft

A2.6 Realizar una calendarización para determinar los tiempos de inicio y cargas de trabajo en el proyecto.

A2.7 Estimar los costos de cada una de las actividades mediante las historias de usuario

Práctica específica 3: Realizar la capacitación.

A3.1 En la negociación y análisis de los requerimientos, determinar el valor de negocio y prioridad en cada caso.

A3.2 Estimar en forma colaborativa el esfuerzo requerido para cada tarea.

A3.3 Realizar “historias de usuario” en las tareas necesarias para lograr un producto satisfactorio.

A3.4 Seleccionar los requerimientos del usuario mediante las “Historias de usuario,” y realizar la capacitación para la conformación de las tareas en la pila del *Sprint*, identificando las actividades que se relacionan con Moprosoft al nivel I.

A3.5 Establecer el mecanismo de desarrollo de la fase de construcción tomando como base a la técnica ágil de TDD.

A3.6 Controlar de los avances de una pila de Sprint, en el uso de las herramientas para: control de los avances del producto, estimación de tiempos y cargas de trabajo, y control del proyecto.

A3.7 Interpretar los datos obtenidos en las tablas de pila de producto y gráfico de producto y tablero de *Scrum*

Práctica específica 4: Generación de reportes

A4.1 Generar reporte de validación de los instrumentos de la capacitación y registrar el histórico de competencias adquiridas.

A4.2 Generar reporte de la calidad de la capacitación y registrar en el historial de lecciones impartidas a los integrantes del equipo.

A4.3 Generar reporte de aprovechamiento de la capacitación individual y registrar en el historial de competencias particulares.

Las actividades del área clave de CAAM, principalmente se basan en el uso de competencias como una mejora constante de la capacidad, para que pueda llevarse a cabo las tareas y responsabilidades asignadas. El objetivo de CAAM es asegurar que todos los individuos tienen las competencias requeridas para realizar sus tareas haciendo uso del área de proceso, mediante un conjunto de prácticas relacionadas que cuando se realizan colectivamente satisfacen la capacidad adquirida en el logro de un nivel de madurez. Las funciones a lograr mediante la aplicación del área de proceso del modelo son:

- Planeación de actividades para ambientes ágiles.
- Comunicación con el cliente del producto
- Uso de herramientas ágiles.
- Uso de herramientas de pruebas.

El flujo de trabajo de CAAM se muestra en la figura 4-5. Usando validaciones dentro de un ciclo, CAAM asegura que la competencia se logra.

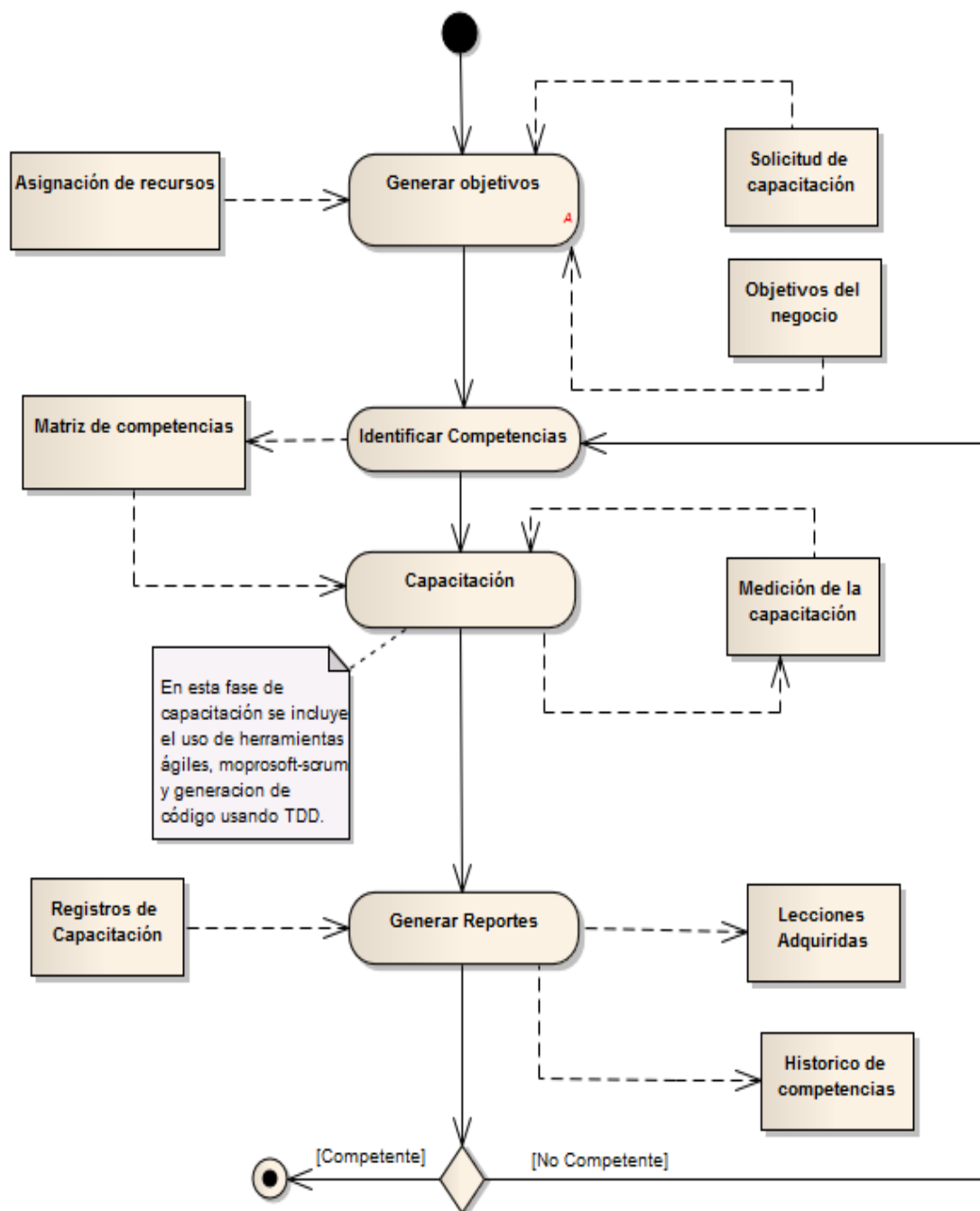


Fig. 4-5 Flujo de CAAM

4.8 Elementos de madurez en CAAM

Como se mencionó anteriormente, CAAM está integrado por niveles de madurez. La tabla 4-1 se muestra las principales funciones asociadas al modelo, y la evolución que presentan a lo largo de los 4 niveles de éste. La tabla 4-2 los indicadores claves

que servirán para identificar el rendimiento obtenido de la aplicación de cada una de las prácticas clave.

Tabla 4-1 Descripción de niveles para la funcionalidad de CAAM.

Función	Nivel 0	Nivel 1	Nivel 2	Nivel 3
Integración Moprosoft-Scrum	No se cuenta con conocimiento	Se inicia con la identificación de las actividades que se realizan en cada una de las fases de Moprosoft cuando se realiza la gestión de un proyecto usando Scrum.	Se inicia el uso de Scrum, cuidando que se cumpla con el mayor número posible de especificaciones de Moprosoft en la categoría de operación.	Se realiza una medición del esfuerzo en el uso de los procesos de gestión, es decir, en el logro de las competencias a este nivel.
Implementación de filosofía ágil	No se cuenta con conocimiento ó se cuenta pero no se aplica	Planear los principales instrumentos ágiles que se usarán en la capacitación.	Establecer una planeación de 40 horas para el uso de: planeación ágil, historias de usuario y estimación de cada uno de los <i>Sprints</i> .	Medición del esfuerzo y competitividad en el uso de la filosofía ágil, en el marco de Moprosoft.
Proceso de construcción de código	No se cuenta con documentación para guiar el proceso de capacitación	Conocimiento de estructuras de tipo TDD	Entrenamiento del desarrollo de código mediante el uso TDD.	Competitividad en la construcción del código mediante TDD. Obtención de métricas de productividad y esfuerzo.

Cabe hacer notar que la medición de las prácticas claves es importante para tener un parámetro que indique de manera cuantitativa si se están logrando el objetivo de CAAM en la tabla 4-2 se marca las métricas que sirven de control del trabajo a implementar.

Tabla 4-2 Indicador de rendimiento de CAAM.

Nivel	Indicadores de rendimiento.
0	Ninguno
1	Estimación precisa de la planeación (EPP), del valor real del cronograma del proyecto. $EPP = \frac{\text{duración real de la capacitación}}{\text{duración estimada de la capacitación}}$

Nivel	Indicadores de rendimiento.
2	Estimación del número total (NTC) de las competencias logradas. $NTC = \frac{\text{Número de competencias logradas}}{\text{Número de competencias planeadas}}$ Porcentaje de cumplimiento de manejo de moprosoft-scrum. PCMS=(Número de actividades identificadas por el alumno/numero de actividades identificadas en moprosoft-scrum)*100 Impacto en conocimiento=empleados que mejoraron el conocimiento/(Total de empleados que recibieron la capacitación relacionada)
3	Índice de gestión: Resultados esperados/ Resultados alcanzados. Participación: Capacitación ejecutada/ Número capacitación programada. Actividad: Número de clases planeadas/ Número de clases dictadas. Impacto en resultados de productividad= Productividad después del proceso de capacitación/productividad antes del proceso de capacitación

4.9 Proceso de implementación de CAAM

En esta sección se describen los pasos que se ejecutan, por medio de actividades concretas, para la implementación de CAAM. Para diseñar este proceso de implementación, se tomó como base la especificación revisada en (Jaap, Heijstek, & Linders, 2008); donde se explica la planificación de la implementación desde el proyecto, producto, integración del proceso y medición. La planeación de proceso desarrolla la capacidad para definir, implementar y mejorar conjuntos de procesos en una organización.

Los pasos a seguir para implementar CAAM son:

Paso 1.- Para la implementación del modelo de entrenamiento propuesto en este trabajo (CAAM) se necesita detectar el grado de conocimiento de la aplicación de las técnicas ágiles por parte de los participantes. Para esto se aplica a los desarrolladores de la empresa un cuestionario inicial, que consta de 11 preguntas, por medio del cual se determina el conocimiento de las prácticas de manera intuitiva o la carencia de las mismas. En el anexo E se muestra este cuestionario inicial.

Paso 2.- Determinación del grado de conocimientos. Aquí se deben identificar las competencias ágiles necesarias a usar en el ambiente de trabajo de la empresa donde está implementándose CAAM.

Paso 3.- Evaluar que tanto hay que avanzar. Implica definir cuantos cursos/horas de trabajo son necesarios y cada cuando se deben de evaluar.

Paso 4.- Definición de la dinámica de los cursos. Esto se realiza mediante la impartición de los materiales por parte del entrenador(a) y resolviendo casos prácticos, teniendo cuidado de considerar las competencias a cubrir. Se realiza una medición del logro de las competencias a nivel individual y en equipo. El seguimiento del curso se efectúa mediante la planeación de las actividades teóricas y prácticas, como se referencia en el anexo G. Esto permite pasar del nivel cero al siguiente (uno) de Moprosoft, debido que en un inicio no se contaba con un modelo de planeación ágil y en el siguiente del nivel del modelo de CAAM se propone como característica principal el planear y el conocimiento de las competencias a lograr.

Paso 5.- A la finalización del curso se contesta un cuestionario de 11 preguntas para determinar la nueva actitud que adopta el/la capacitado(a) en el desarrollo de software. Este cuestionario mide la adquisición de habilidades asociadas a: trato al cliente, tamaño del equipo, cumplimiento de las planeación, tiempos de decisión. El Anexo F detalla al cuestionario final. Asimismo, la tabla F.1 detalla la habilidad que está midiendo cada pregunta.

Paso 6.- Hacer reuniones con los equipos de trabajo para “hacer suyos” los estilos/actividades, ajustar los roles de las personas de acuerdo a lo aprendido en los cursos tomados.

Paso 7.- Ajustar el plan de cursos realizados en el paso 3 de acuerdo a la filosofía de CAAM.

Capítulo 5 RESULTADOS DE LA INVESTIGACIÓN

5.1 Resultados: análisis de datos

De la hipótesis planteada en la presente tesis el elemento a probar es “mejorar las competencias de los desarrolladores” mediante un modelo de procesos de capacitación. Para esto se efectuaron tres cursos con un total de 32 asistentes de los cuales para fines del reporte solamente se tomaron los resultados emitidos por 27, debido a la inconsistencia de los datos restantes. El seguimiento del curso se fundamentó en el proceso propuesto en la sección 4.9. Siguiendo los pasos en primer punto se establece la realización de un mecanismo para detectar el grado de conocimiento del uso de las técnicas ágiles por parte de los participantes. Mediante la aplicación de un cuestionario de 11 preguntas como fase de pre-capacitación y la misma cantidad de preguntas se realizaron en una siguiente de post-capacitación para verificar el grado de aceptación por parte de los participantes.

La finalidad de ambas encuestas fue el encontrar la asociación de las respuestas a las características principales de las competencias que se desean desarrollar, que son: la planeación de actividades para ambientes ágiles, la comunicación con el cliente, el uso de herramientas ágiles y el uso de herramientas de pruebas.

5.2 Resultados pre-capacitación.

En esta sección se hace un análisis de los resultados obtenidos en un primer curso y se recabo la información en el instrumento que se muestra en el anexo E. El resultado de personas que respondieron a la solicitud de la encuesta fue de 26 participantes en el curso. De éstos, se obtuvo un total de 275 reactivos contestados.

5.2.1 Análisis de confiabilidad de las encuestas de pre-capacitación.

Para determinar la confiabilidad del instrumento de medición se utilizó el coeficiente denominado *Alfa de Cronbach* (Cronbach, 1984) por la facilidad de su aplicación. Éste es un índice de consistencia interna que toma valores entre 0 y 1; sirve para comprobar si el instrumento que se está evaluando recopila información

defectuosa y por tanto nos llevaría a conclusiones equivocadas, o si se trata de un instrumento fiable que hace mediciones estables y consistentes, en este trabajo se realizó el cálculo del coeficiente haciendo uso de la hoja de cálculo de *Excel* a partir de las varianzas de los datos recabados se hizo uso de la siguiente formula.

$$\alpha = \left[\frac{k}{k+1} \right] \left[1 - \frac{\sum_{i=1}^k S_i^2}{S_t^2} \right]$$

S_i^2 es la varianza del dato i

S_t^2 es la varianza de los valores totales observados y

k es el numero de preguntas.

Para la primer encuesta el alpha Cronbach fue de 0.86, lo cual indica que el instrumento esta validado para obtener información confiable de sus datos obtenidos. Se observa que el resultado de aplicar el instrumento mediante una escala de Likert , herramienta que sirve como escala en encuestas de investigación, en el experimento realizado al cuestionario de pre-capacitación existió una concentración de los datos sesgada a de acuerdo.

5.2.2 Descripción de las características de pre-capacitación.

Las características mostradas en las secciones siguientes describen las relaciones entre las preguntas y las competencias que se planearon como parte de la hipótesis de investigación. Mediante la agrupación de las variables se establecen los índices de aceptación por parte de los encuestados.

5.2.3 Planeación de actividades para ambientes ágiles pre-capacitación.

La planeación de actividades para la validación como rasgo del entrenamiento se relacionó con las variables que se muestran en la tabla 5-1. El porcentaje para esta característica es del 89.5% lo que nos representa un conjunto de personas que están "Ni en desacuerdo ni en acuerdo", con el uso de éste tipo de tareas el promedio total muestra que los participantes tienen una tendencia a hacia los valores altos de esta escala.

Tabla 5-1 Variables que integran la Planeación de actividades pre-capacitación.

Variable	Media	Desviación Estándar
Entrega de funcionalidad	3.24	1.0116
Aceptación de cambios por parte del equipo	3.52	1.1944
Logro de trabajo entregado	3.48	0.9626
Utilización de las metodologías	2.96	1.0985
Promedio	3.3	1.066775

5.2.4 Comunicación con el cliente pre-capacitación.

Otro de los rasgos importantes que se consideran en la metodología ágil es la constante comunicación con el cliente. En la tabla 5-2 se muestran los resultados de las dos variables que se relacionan a esta categoría, en este rasgo las personas se muestran en un promedio de 89.86% lo que muestra una tendencia a valores altos de la escala.

Tabla 5-2 Variables que integran la Comunicación con el cliente pre-capacitación.

Variable	Media	Desviación Estándar
Satisfacción de los clientes	3.40	1.1180
Comunicación con los <i>stakeholders</i>	3.52	1.0847
Promedio	3.46	1.10135

5.2.5 Uso de herramientas ágiles pre-capacitación.

La siguiente agrupación de las variables a considerar es el uso de las características ágiles, mediante la agrupación de las variables que se muestra en la tabla 5-3. Los resultados identifican que los participantes tienen una tendencia a trabajar de manera ágil de manera intuitiva, sin tener aparente conocimiento previo de las metodologías ágiles. Las personas que están "algo de acuerdo" o "totalmente de acuerdo" es un promedio es de un 90.84%. es decir, debido a la media obtenida se tienen una tendencia de que las personas opten por los valores superiores de la escala.

Tabla 5-3 Variables que integran el uso de herramientas ágiles pre-capacitación.

Variable	Media	Desviación Estándar
Construcción de software de manera conjunta	3.64	1.0755
ambiente de trabajo adecuado	3.84	0.7461
Medición del progreso del desarrollo	3.56	1.0033
Organización del equipo	3.52	0.9183
Promedio	3.64	0.9358

5.2.6 Uso de herramientas de pruebas pre-capacitación.

Esta clasificación formada por una sola variable también muestra una tendencia hacia la aceptación por parte de los encuestados, como se muestra en la tabla 5-4. El porcentaje para esta característica muestra que las personas encuestadas tienden hacia las opciones de la escala de Likert “algo de acuerdo” o “totalmente de acuerdo” en un promedio del 93.86%. Lo que puede indicar un uso de pruebas antes de codificar tiene un fuerte arraigo en los participantes en su quehacer de la construcción del software.

Tabla 5-4 Variable que muestra el Uso de herramientas de pruebas pre-capacitación.

Variable	Media	Desviación Estándar
Efectividad del mecanismo de pruebas	3.52	0.8718

5.3 Resultados post-capacitación.

El siguiente apartado describe el resultado de las encuestas post-capacitación, ver anexo F, realizadas a las tres semanas del taller de *Scrum*, con la intención de que se adoptará los conceptos y mecanismos de trabajo de este esquema en las empresas con certificación de Moprosoft que participaron en el experimento de la tesis. El resultado de personas que respondieron a la solicitud de la encuesta fue de 17, de los 26 participantes en el curso. De éstos, se obtuvo un total de 198 reactivos contestados.

Mediante la siguiente descripción de la estadística descriptiva, estudio de frecuencias, obtendremos un resumen de la información donde se puede observar la distribución de las ocasiones que se selecciona una respuesta en la aplicación del segundo cuestionario (post-capacitación).

5.3.1 Descripción de las características de post-capacitación.

En las siguientes secciones se muestran las agrupaciones de variables correspondientes a las características de la capacitación implementadas en la tesis. La redacción fue diferente a las del cuestionario de pre-capacitación, sin embargo el significado se relaciona en cada una de las características de las competencias a lograr.

5.3.2 Planeación de actividades para ambientes ágiles post-capacitación.

El porcentaje de aceptación de esta característica por parte los entrevistados que se encuentran entre "algo de acuerdo y "totalmente de Acuerdo" después de tomar el curso, es del 94.28%, debido a que el promedio total de las medias tiende hacia los valores más alta de la escala de Likert.

Tabla 5-5 Variables que integran la planeación de actividades post-capacitación.

Variable	Media	Desviación Estándar
Entrega de funcionalidad	3.944	0.7253
Aceptación de cambios por parte del equipo	4.056	0.8023
Organización del equipo	3.778	1.0602
Efectividad del mecanismo de pruebas	3.167	0.9851
Promedio	3.7362	0.8932

5.3.3 Comunicación con el cliente post-capacitación.

Este rasgo está conformado por las variables que se muestran en la tabla 5-6. El porcentaje de aceptación de los entrevistados que se encuentra entre "algo de acuerdo" y "totalmente de acuerdo" después de tomar el curso de Scrum es de 93.75%.

Tabla 5-6 Variables que integran la comunicación con el cliente post-capacitación.

Variable	Media	Desviación Estándar
Satisfacción de los clientes	4.06	0.9375
Comunicación con los <i>stakeholders</i>	3.778	0.6467
Logro de trabajo entregado	3.5	1.2485
Promedio	3.7793	0.9442

5.3.4 Uso de herramientas ágiles post-capacitación.

La tabla 5-7 muestra los rasgos para la variable de uso de herramientas ágiles. El 93.5% de los entrevistados post-capacitación respondieron "algo de acuerdo" o "totalmente de acuerdo."

Tabla 5-7 Uso de herramientas ágiles de post-capacitación.

Variable	Media	Desviación Estándar
Construcción de software de manera conjunta	3.944	0.9375
Planeación de actividades ágiles	4.111	0.7583
Ambiente de trabajo adecuado	3.5	1.2485
Promedio	3.8516	0.9814

5.3.5 Uso de herramienta de pruebas post-capacitación.

El único rasgo considerado por el cuestionario aplicado al uso del manejo de herramienta de pruebas por parte de los encuestados tuvo un porcentaje de 84.33%, es decir, los entrevistados se encuentran entre Algo de Acuerdo y Totalmente de Acuerdo.

5.4 Análisis de resultados de los cuestionarios pre- y post-capacitación.

Se observa en la tabla 5-8 la mejora en algunos de los rasgos de las competencias a adquirir mediante el curso de *Scrum*. En el caso de la característica de planeación de actividades para ambientes ágiles se observa una mejora del 4.78% puntos con respecto al primer cuestionario. De manera similar, en la característica

"comunicación con el cliente" se tiene una mejora del 3.89% puntos y en la de "uso de herramientas ágiles" una de 2.66% puntos, al compararse el cuestionario aplicado antes de la capacitación con respecto al aplicado después de capacitación.

Por último puede notarse que el rasgo de "probar en ambiente ágiles" es un concepto que no tuvo una mejora, sino al contrario, desciende en un 9.53% puntos. Esto indica que es necesario desarrollar una mejor estrategia al enseñar el concepto de "probar primero," comparado al tradicional esquema "codificar-probar". Esto es una buena oportunidad para realizar el paso número 6 del proceso de CAAM, en el cual se indica "Hacer reuniones con los equipos de trabajo para "hacer suyos" los estilos/actividades, ajustar los roles de las personas de acuerdo a lo aprendido en los cursos tomados" y poder determinar cuáles son los conceptos no adquiridos.

En resumen, y analizando estos resultados, se puede establecer que el modelo CAAM cumple de manera parcial a los intereses de capacitación ágil en empresas con modelos de control de procesos o no teniéndolo, mediante el seguimiento de cada uno de los procesos establecidos en la sección 4.9.

Tabla 5-8 Diferencia porcentual de competencias post versus pre capacitación.

Competencia	Post-Capacitación	Pre-Capacitación	Diferencia
Planeación de Actividades para ambientes ágiles	89.50%	94.28%	4.78%
Comunicación con el cliente	89.86%	93.75%	3.89%
Uso de herramientas ágiles	90.84%	93.50%	2.66%
Uso de herramientas de prueba	93.86%	84.33%	-9.53%

Capítulo 6 CONCLUSIONES

El objetivo de esta investigación fue aportar un nuevo modelo de mejora del proceso de capacitación para desarrolladores de software en PYMES basado en ambientes ágiles, llamado CAAM. Este modelo permite a las empresas realizar un conjunto de actividades para lograr la capacidad en el crecimiento de la formación de

personal competente en tiempos cortos, enfocándose en rasgos de planeación, comunicación con el cliente, uso de herramientas ágiles y control de pruebas de software. La importancia del modelo de CAAM radica principalmente en que, si es utilizado correctamente, permitirá la maduración de las empresas desarrolladoras de software en el campo de la capacitación. Es importante recalcar que, la revisión de diversos modelos de madurez publicados en el estado del arte y llevado a cabo durante la presente investigación, dio origen al modelo de capacitación sobre modelos ágiles presentado aquí. Básicamente, CAMM fue inspirado en el Modelo de Madurez de Capacidad desarrollado por el *Software Engineering Institute* conocido como CMMI .

Durante el desarrollo de esta investigación se identificaron las asociaciones entre actividades propias a los procesos de la norma Moprosoft en los procesos “*Administración de proyectos específicos*” y “*Desarrollo y mantenimiento de Software*”, los cuales se emparejaron a la gestión de proyectos ágiles tipo *Scrum*. Para obtener las principales características a considerar en CAAM se realizó un comparativo de las principales tareas donde se muestra una compatibilidad en ciertas funciones. De esta manera se respetó la esencia de Moprosoft en el sentido del manejo de la documentación.

Posteriormente, se obtuvieron las competencias basadas en la comparación de Moprosoft y *Scrum*. Con estas competencias, y siguiendo el modelo de implementación de CAMM, también definido en esta investigación, se diseñó un curso, el cual fue aplicado a 8 empresas en la ciudad de Puebla, México. A fin de medir la validez del modelo, se diseñaron 2 cuestionarios a aplicarse antes y después de presentar el curso a las empresas. El análisis estadístico que se realizó sobre los resultados de estos cursos mostró evidencia de que el modelo sirvió para mejorar la capacitación de los desarrolladores de software de PyMEs, en la mayoría de los rubros que fueron evaluados. Este tipo de capacitaciones permitirá a las empresas aminorar los tiempos de capacitación, adentrándose en lo esencial que los

participantes deben de conocer para llevar las habilidades adquiridas a la práctica de manera pronta.

Hasta la fecha de terminación de esta tesis, no se habían encontrado trabajos donde se tratara de manera profunda el estudio de modelos de mejora de la capacitación para las empresas PyMEs mexicanas de desarrollo de software. El área de estudio de esta investigación presenta buenas oportunidades de desarrollo desde las perspectivas de mejora en el proceso, tasas de retorno sobre la inversión, estrategias de adopción del conocimiento y aplicación del mismo en tiempos cortos que no impliquen altos costos de capacitación.

6.1 Trabajo futuro

De acuerdo a los resultados encontrados al aplicar el curso y descritos en el capítulo 6, se encontró que se debe ajustar la manera en cómo se lleva a cabo el proceso de “uso de pruebas” durante el diseño e impartición del curso inicial diseñado usando CAMM y utilizado para medir la validez del modelo. Esto se deduce que el análisis de resultados realizado presentó una fuerte diferencia porcentual negativa en dicho rubro.

Es importante considerar que el hecho de implementar la filosofía ágil desde las áreas académicas es un reto, ya que implica un cambio en el paradigma de la construcción de aplicaciones. Por esta razón, se propone utilizar el modelo para incluir algunos cursos de capacitación corta en la formación de especialistas en ingeniería de software a nivel licenciatura.

Bibliografía

- Albaladejo, X. (17 de 01 de 2010). *proyectosagiles.org*. Recuperado el 20 de 07 de 2011, de proyectosagiles.org: <http://www.proyectosagiles.org/lista-requisitos-priorizada-product-backlog>
- Alfonso, M., & Botío, A. (2005). *Resources for Agile Software Development in the Software Engineering Course*. Alicante: Proceedings of the 18th Conference on Software Engineering Education & Training.
- Barbazette, J. (2006). *Training needs assesment*. Portland: Pfeiffer.
- Beck, E. (2003). *Test-Driven Development By Example*.
- Beck, K., Beedle, M., Bennekum van, A., & Cockburn, A. (15 de 01 de 2001). *Manifiesto por el Desarrollo Ágil de Software*. Recuperado el 2011 de 04 de 03, de Manifiesto por el Desarrollo Ágil de Software: <http://www.agilemanifesto.org/iso/es/manifesto.html>
- Beth, M., Konrad, M., & Shrum, S. (2009). *Guía para la integración de procesos y la mejora de productos*. Pearson Education.
- Bologa, R., & Lupu, A. R. (2007). Accelerating the Sharing of Knowledge in Order To Speed Up the Process of Enlarging Software Development Teams - a Practical Example. *Proceedings of the 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases.*, 90-95.
- Buckley R. Caple J. (2009). The theory & practice of training. En R. B. Caple Jim, *The theory & practice of training* (págs. 24-33). British: Kogan Page.
- Bushnell, D. (1990). Input, process, output: A model for evaluating. *Training and Development Journal*, 41-43.
- Carnegie, M. U. (1995). *The Capability Maturity Model: Guidelines for Improving the Software Process*, Reading. Massachuset: Addison-Wesley.
- Catalano A. M.; Avolio de Cols S.; Sladogna M. (2004). *Diseño curricular basado en normas de competencia laboral: conceptos y orientaciones metodológicas*. Uruguay: Banco InterAmericano de Desarrollo.
- Centeno-Tellez, A., & Gómez-Gil, P. (08 de 08 de 2010). *Publicaciones P. Gómez-Gil*. Recuperado el 05 de 04 de 2012, de Publicaciones P. Gómez-Gil: <http://ccc.inaoep.mx/~pgomez/publications/congress/PggAct10.pdf>
- Checkland P. & Scholer J. (1990). *Soft Systems Methodology in Action*. New York: John Wiley & Sons.
- Chuan-Hoo, T., & Week-Kek, T. (2010). Designing an Information Systems Development Course to Incorporate Agility, Flexibility, and Adaptability. *Communications of the Association for Information Systems*, 172-192.
- CMMI. (2010). *CMMI for Development, Version 1.3*. Carnegie Mellon University: Software Engineering Process Management Program.
- Cockburn, A. (2000). *Agile Software Development*. Recuperado el 15 de 07 de 2012, de <http://www.snip.gob.ni/xdc/Agile/AgileSoftwareDevelopment.pdf>
- Cockburn, A. (2005). *Crystal/Clear: A Human-Powered Methodology for Small Teams*. New York: Addison-Wesley.
- Conradi, R., & Fuggeta, A. (2002). Improving Software Process Improvement. *IEEE Software*, 92-99.

- Cronbach, L. (1984). *Essential of psychological testing*. New York: Harper & Row Publishers.
- Cuellar C. (2010). Valores y Capacitación. *Software gurú*, 12-13.
- Dawson, R. (2000). Twenty dirty tricks to train software engineers. *Software Engineering, 2000. Proceedings of the 2000 International Conference*, 209 - 218 .
- Dedrick J., Kraemer K. & Palacios J. (1 de January de 2001). Impacts of Liberalization and Economic Integration on Mexico's Computer Indutry. California, Irvine, EEUU.
- Dingsøyr, T., Nerur, S., Balijepally, V., & Brede, N. (2012). A decade of agile methodologies: Towards explaining agile software development. *The Journal of Systems and Software*, 1213-1221.
- Dubinsky, Y., & Hazzan, O. (2003). Extreme programming as a framework for student project coaching in computer science capstone courses. *Proceedings of the first IEEE International Conference on software*, 53-59.
- Dueholm, S. (2010). Software Process Improvement as organizational change: A metaphorical analysis of the literature. *Elsevier Science Inc.*, 2128-2146.
- Escalante, V., & Seis, E. (2003). *Sigma Metodología y técnicas*. México: Limusa.
- Flores, B., Astorga, M. A., Olguín, J. M., & Andrade, M. C. (2008). Experiences on the Implementation of MoProSoft and Assessment of Processes under the NMX-I-059/02-NYCE-2005 Standard in a Small Software Development Enterprise. *Mexican International Conference on Computer Science*, 323-328.
- Fraga A. & Llorens J. (2007). Training Initiative for New Software/Enterprise Architects: An Ontological Approach. *Proceedings of the Sixth Working IEEE/IFIP Conference on Software Architecture*, 19.
- Garg, K., & Varma, V. (2008). Software Engineering Education in India: Issues and Challenges. . *Software Engineering Education and Training, 2008. CSEET '08. IEEE 21st Conference*, 110-117.
- Glacer, H. (2008). Software Engineering Process Management. *Technical Note CMU/SEI-2008-TN-003*, 1-38.
- Goldstein, I. L. (1993). *Training in organizations: Needs Assesment, Development, and Evaluation*. Pacific Grove, CA: Brooks/Cole.
- Gresse von Wangenheim, C., Punter, T., & Anacleto, A. (2003). Software measurement for small and medium enterprise. A Brazilian-German view in extending the GQM method., (págs. 1-7).
- Hammer M. & Champy J. (1993). *Reengineering the Corporation*. New York: HarperBusiness Essential.
- Hareton, L., & Terence, Y. (2001). A process framework for small projects. *Software process: Improvement and practice*, 67-83.
- Harrison, N., & Foote, B. (2000). Pattern Languages of Program Design 4. En M. Beedle, *SCRUM: An Extension Pattern Language* (págs. 637-651). Massachusset: Addison-Wesley.
- Hernández, S., Fernández, C., & Baptista, P. (2007). *Metodología de la Investigación*. México: McGraw-Hill.

- Herrera, E., & Trejo, A. (2003). A methodology for self-diagnosis for software quality assurance in small and medium-sized industries in Latin America. *The electronic journal on Information Systems in Developing Countries*, 1-13.
- Huang P., Chen M. & Chen S. (2008). The practice training in the Software Engineering Education. *The 9th International Conference for Young Computer Scientists*, 2636-2640.
- Humphrey, W. (1989). *Managing the Software Process*. Reading, MA.: Addison-Wesley.
- IEEE1074-97. (1997). *IEEE Std 1074-1997, IEEE Standard for Developing Software Life Cycle Processes*.
- Jaap, C., Heijstek, A., & Linders, B. (2008). *CMMI Roadmaps*. Carnegie Mellon University.
- Janzen D. (2008). Does Test-Driven Development Really Improve Software Design Quality? *Published by the IEEE Computer Society*, 77-84.
- Kadjo M., Forssander S. & Olsson U. (2001). Corrective Maintenance Maturity Model Maintainer's Education and Training. *Proceedings of the 23rd International Conference on Software Engineering*, 12-19.
- Kirkpatrick D. (2009). *Evaluating Training Programs: The four levels*. San Francisco, California: Berrett-Koehler.
- Klappholz, D., Bernstein, L., & Port D, D. (2003). Tools for Outcomes Assessment of Education and Training in the Software Development Process. *Proceedings of the 16th Conference on Software Engineering Education and Training*, 331-332.
- Li, P., Liu, Z., & Guo, X. (2008). Research on the Diversification Training Mode of Software Talents Based on University-Enterprise Cooperation. *International Conference for Young Computer Scientists*, 2417-2421.
- Lutteroth C., Luxton A, Dobbie G. & Hamer J. (2007). A maturity model for computing education. *Proceedings of the Ninth Australasian Conference on Computing Education*, 107-114.
- Mark P., Bill C. & Beth C. (1993). Capability Maturity Model Version 1.1. *Software Engineering Institute*, 18-27.
- Marsall S. & Mitchel G. (2006). An E-Learning Maturity Model Version Two. *Informing and Guiding E-Learning Architectural Change and Development Project Report*, 132.
- Martínez, A. R., Dewhurst, F., & Dale, B. G. (1988). Total quality management: origins and evolution of the term. *The TQM Magazine*, 378-386.
- McCabe T. (1976). A complexity measure. *Proceedings of the 2nd international conference on Software engineering*, 407.
- Mejía, M., Ania, I., & Gamboa, R. (31 de 12 de 2006). *Diagnóstico de la Industria de Servicio de Software en México*. Recuperado el 17 de 04 de 2011, de AIS Electronic Library (AISeL): <http://aisel.aisnet.org/amcis2006/497/>
- Nagappa, N., & Maximilie, E. M. (2008). Realizing quality improvement through test driven development: results and experiences of four industrial teams. *IEEE papers-software engineering*, 289-300.

- Nava, D. (2009). *Representación de procesos para la industria de software (MOPROSOFT) en el metamodelo de ingeniería de procesos, software y sistemas (SPEM) 2.0*. México: UNAM Posgrados.
- Niazi, M. (2005). A maturity model for the implementation of software process improvement: an empirical study. *Elsevier Science Inc.*, 155-172.
- Nokes, S., & Kelly, S. (2007). *The Definitive Guide to Project Management: The Fast Track to Getting*. Glasgow:Great Britain: Pearson Education Limited.
- Normalización y Certificación Electrónica A.C. (2007). "*Tecnología de la Información Software Modelos de procesos evaluación para desarrollo y mantenimiento de Software. Parte 1: Definición de Conceptos y Productos*". Normalización y Certificación Electrónica A.C. Recuperado el Agosto de 2010, de <http://www.nyce.org.mx/>
- Oktaba, H., Alquicira, C., Su, A., & Martínez, A. (2005). *Modelo de procesos para la Industria del Software. Por niveles de capacidad de procesos*. México: Secretaría de Economía.
- Oktaba, H., García, F., Piattini, M., & Pino, F. (2006). MoProSoft: A Software Process Model for Small Enterprises. *Proc. 1st Int'l Research Workshop for Process Improvement in Small Settings, special report CMU/SEI-2006-SR-001.*, 93-101.
- Oktaba, H., García, F., Piattini, M., & Ruiz, F. (2007). Software Process Improvement: The Competisoft Project. *IEEE Computer Society*, 21-28.
- Olivos, F. (2010). Comparativa de equipos de trabajo mexicanos ante los métodos . *VIII Congreso Internacional sobre Innovación y Desarrollo Tecnológico*, 457-463.
- PMI, P. M. (2004). *A guide to the project management body of knowledge*.
- Prochazka, J., Kokott, M., Chmelar, M., & Krchnak, J. (2011). Keeping the spin – from idea to cash in 6 weeks. *International Conference on Global Software Engineering*, 124-130.
- Prosoft. (2002). *Programa para el desarrollo de la industria de software*. México: Secretaría de Economía.
- Radziwill N. (2005). An ethical theory for the advancement of professionalism in software engineering. En C. S. T., *Fundamental Concepts for the Software Quality Engineer* (págs. 15-28). Milwaukee: American Society for Quality.
- Rassmusson J. (2010). *The agile Samurai, How Agile Masters Deliver Great Software*. Dallas: Susannah Davison Pfalzer.
- Reichlmayr, T. (2003). The Agile Approach in an Undergraduate Software Engineering Course Project. *Proceedings of the 33rd ASEE/IEEE Frontiers in Education Conference*, 5-8.
- Rising, L., & Janoff, N. S. (2000). The Scrum Software Development Process for Small Teams. *IEEE Software*, 2-8.
- Rivera, G., & Montero, E. (2004). *Mapeo de CMMI Nivel 2 con MoProSoft*. México: internal report, Mexican Ministry of Economy.
- Rodriguez, G. (2006). Aprendizaje y cultura laboral entre los programadores de Software de Mexico. *I Congreso Iberoamericano de Ciencias, Tecnología, Sociedad e Innovación* , 12-13.

- Rogers, A. W., Campbell, H. R., & Pak, R. (2001). A systems approach for training older adults to use technology. En N. Charness, D. C. Parks, & B. A. Sabel, *Communication Technology and Aging: Opportunities and challenges for the future* (págs. 187-202). New York: Springer Publishing.
- Ruvalcaba, M. (2005). Procesos de Software. *Software Guru*, 4-7.
- Schwaber K. & Beedle M. (21 de 08 de 2011). *The Scrum Guide*. Recuperado el 21 de 04 de 2011, de The Scrum Guide: http://www.scrumalliance.org/scrum_student_resources
- Schwaber, K., & Beedle, M. (2001). *Agile Software development with Scrum*.
- Spada, D. (2005). *Usabilidad en el proceso de desarrollo de SCRUM*. Tesis Doctoral. SPICE. (14 de 04 de 2011). *El portal de la norma ISO 15504 en castellano*. Recuperado el 11 de 05 de 2011, de El portal de la norma ISO 15504 en castellano: <http://www.iso15504.es/>
- SPIRE. (01 de 01 de 1999). *Software Process Improvement in Regions of Europe* . Recuperado el 19 de 05 de 2011, de Software Process Improvement in Regions of Europe : <http://www.cse.dcu.ie/spire/>
- Stapleton, J. (1997). *Dynamic Systems Development Method: The Method in Practice*. Great Britain: Addison-Wesley.
- Stephenson, W. (1953). *The study of behavior, Q technique and its methodology*. Chicago: University of Chicago.
- Sterns, H. L., & Doverspike, D. (1989). Aging and the training and learning process in organization. En Golstein, *Training and development in work organizations: Frontiers of industrial and organizational psychology* (págs. 299-332). San Francisco, CA: Jossey Bass.
- Steve D. (2010). Learning, Training and Development in Organizations. En *Learning, Training and Development in Organizations* (págs. 67-95). New York: Routledge.
- Sull F., Melnik G., Turhan B., Layman L., Diep M. and Erdogmus H. (2010). What Do We Know about Test-Driven Development? *IEEE Computer Society*, 16-19.
- Vargas, A., Olguín, J. M., & Flor, B. L. (2006). Caracterización de los factores de cambio para la implantación de MoProSoft en las PyMES. *Congreso Internacional de Ciencias Computacionales (CICOMP)*, 13-17.
- Veenendaal, E. (2010). *Test Maturity Model integration*. Ireland: TMMi Foundation.
- Visconti M. & Cook C. (1998). Evolution of a maturity model critical evaluation and lessons learned. *SpringerLink*, 223-237.
- VisionOne. (28 de 06 de 2010). *Agile development: Results Delivered*. Recuperado el 17 de 05 de 2011, de Agile development: Results Delivered: http://www.versionone.net/pdf/AgileDevelopment_ResultsDelivered.pdf
- Watss H. (1988). Characterizing the Software Process: A Maturity Framework. *IEEE Software*, 73-79.
- Wingreen, S., & LeRouge, C. (2009). Structuring Training for IT Professionals and the Firm: An Application of the Q-Methodology. *IJGMS*, 53-67.

Anexo A. Cuestionario inicial para empresa

Aquí se presenta el cuestionario utilizado para determinar el conocimiento del grado de adopción de las metodologías ágiles en la industria del software en México. Se aplica este cuestionario pues a la fecha de publicación de este trabajo no existen estadísticas formales sobre el estado que guarda la industria de software mexicana en el conocimiento y aplicación de las metodologías ágiles. Este cuestionario es aplicado a través de Internet, y está disponible en la página: (www.upt.edu.mx)

Objetivo del cuestionario: Conocer el grado de adopción de las metodologías ágiles para el incremento de la productividad en el desarrollo profesional de los individuos en una organización.

La herramienta deberá de dar respuesta a las siguientes preguntas:

¿Qué metodologías y prácticas ágiles están siendo adoptadas en la industria del software mexicana?

¿Cómo se están adoptando las metodologías ágiles en la industria de software mexicana?

¿Cuáles son las diferencias en términos de los beneficios obtenidos, retos y limitaciones durante la adopción de metodologías ágiles en las compañías mexicanas?

¿Cuántas empresas mexicanas cuentan con la implementación de alguna metodología ágil, además de haber obtenido la verificación de moprosoft al menos a nivel 1?

Cuestionario aplicado.

1.- ¿Cuál es el estado o entidad federativa donde se encuentra tu centro de trabajo de desarrollo de Software?

2.- ¿Qué giro tiene tu organización?

Desarrollo de software empaquetado

Desarrollo de software de sistema y herramientas para desarrollo de software aplicativo
Desarrollo de software aplicativo
Desarrollo de software embebido
Servicios de consultoría de software
Servicios de análisis de sistemas computacionales
Servicios de diseño de sistemas computacionales
Servicios de programación de sistemas computacionales
Servicios de procesamiento de datos
Servicios de diseño, desarrollo y administración de base de datos
Servicios de implantación y pruebas de sistemas computacionales
Servicios de integración de sistemas computacionales
Servicios de procesamiento de datos
Servicios de seguridad de sistemas computacionales y procesamiento de datos
Otros

3.- ¿Con que nivel de Moprosoft cuenta tu empresa?

Ninguno

1

2

3

4

5

4.- ¿Mercado al que ofrecen sus servicios?

Gobierno

Manufactura/Extracción

Comercio/Distribución

Informática/Telecomunicaciones

Ninguno

5.-¿Qué rol tienes en la organización?

Desarrollador

Empresario

Evaluador

Gestor de Base de Datos

Gestor de Procesos
Gestor de Proyectos
Gestor de TI
Personal de Operaciones/Soporte
Otros

6.-¿Aplicas metodologías ágiles en tu organización?

Siempre
Casi siempre
Algunas Veces
Nunca

7.- ¿Aplicas prácticas ágiles en tu organización?

Siempre
Casi siempre
Algunas Veces
Nunca

8.-¿Cuándo surge la necesidad de utilizar la filosofía ágil en tu empresa?

Nuevo proyecto
Consultoría
Interés personal/equipo
Soporte de herramientas
Nunca

9.-¿ Cuánto experiencia tienes con el uso del desarrollo ágil?

Nunca lo he usado
Menos de un año
1 a 5 años
5 a 10 años

10.-¿ En qué fase del desarrollo de software aplicas la filosofía ágil?

Planeación
Desarrollo
Pruebas
En todo
En ninguna

11.- ¿Dónde haces uso de la filosofía ágil?

En la organización donde laboro
En un equipo de trabajo donde participo
De manera individual
En ninguna parte

12.- ¿Qué metodologías ágiles usas en tu trabajo de desarrollo de software?

- | | | |
|--|--|--|
| ☐ Scrum | ☐ DSDM | ☐ Personalizada |
| ☐ XP | ☐ FDS | ☐ Ninguna |
| ☐ Híbrido Scrum | ☐ Desarrollo Lean | |

13.- ¿Qué practicas ágiles usas en tu trabajo de desarrollo de software?

- | | | |
|---|---|--|
| ☐ Deliberación frecuente incremental de software | ☐ Desarrollo dirigido por pruebas (TDD) | ☐ Reuniones diarias |
| ☐ Integración Continua | ☐ Participación de los <i>Stackholders</i> | ☐ Otras |
| ☐ Planificación de versiones | ☐ Pruebas unitarias | |

14.- ¿Qué nivel de satisfacción has encontrado con el uso de metodologías ágiles en tu organización?

- ☐ Excelente
☐ Buena
☐ Regular
☐ Mala

15.- ¿Qué impacto has detectado en tu organización con el uso del desarrollo ágil en comparación con el desarrollo tradicional?

- | | | |
|--|---|---|
| ☐ Rentabilidad de la empresa | ☐ Temprana detección de errores y defectos | ☐ Previsión de problemas |
| ☐ Trabajo en equipo, comunicación, y cooperación del equipo de desarrollo | ☐ Ritmo sostenido y equilibrado de la carga de trabajo | ☐ Ninguno
No las uso |
| ☐ Productividad en el equipo de desarrollo | | |

16.- ¿Cómo percibes la productividad del desarrollo de software al usar la filosofía ágiles en comparación al desarrollo tradicional?

- He tenido bastante mejora
- He tenido alguna mejora
- No he notado impacto
- Noto la productividad mucho peor
- No aplica

17.- ¿Cómo percibes el esfuerzo de desarrollo de software al usar la filosofía ágil en comparación al desarrollo tradicional?

- Encuentro que se requiere menor esfuerzo
- He tenido alguna disminución en el esfuerzo
- No he notado impacto en el esfuerzo
- Encuentro que se requiere mayor esfuerzo
- No aplica

Anexo B. Comparación de Roles *Scrum* y Moprosoft en su proceso de “Administración de Proyectos Específicos”

En este anexo se presenta un análisis de comparación entre roles y actividades definidos en Moprosoft y *Scrum* para el proceso de “Administración de Proyectos Específicos”. La tabla B.1 compara roles; la tabla B.2 compara actividades de planificación; la tabla B.3 compara actividades de realización; la tabla B.4 compara actividades de cierre.

Tabla B.1. Comparación de roles Moprosoft y *Scrum*

Rol de Scrum	Rol de Moprosoft
<i>Scrum Master</i>	Responsable de Gestión de Proyectos (RGPY)
<i>Scrum Master</i>	Responsable de la administración del proyecto específico (RAPE)
Cliente	Cliente (CL)
<i>Scrum Master</i>	Responsable del subcontrato (RSC)
<i>Scrum Master</i>	Responsable de desarrollo y mantenimiento de software (RDM)
Equipo	Equipo (ET)

**Tabla B.2. Comparación de Actividades entre Moprosoft y *Scrum*:
fase de Planificación**

Rol Moprosoft	Descripción Moprosoft	Rol Scrum	Artefactos Scrum	Fase de Scrum
Planificación				
RGPY RAPE RDM	A1.1 Revisar con el responsable de gestión de proyectos la descripción del proyecto.	<i>Scrum Master</i>	<i>Product Backlog</i>	Pre-Juego
RAPE CL	A1.3 Definir conjuntamente con el cliente el protocolo de entrega de cada uno de los entregables especificados en la descripción del proyecto.	<i>Scrum Master</i> <i>Product Owner</i>	<i>Product Backlog</i>	Pre-Juego
RAPE	A1.4 Identificar el número de ciclos y las actividades específicas que deben llevarse a cabo para producir entregables y sus componentes identificados en la Descripción del proyecto. Identificar las actividades para llevar a cabo el Protocolo de Entrega. Documentar el resultado como Ciclos y Actividades.	<i>Scrum Master</i> Equipo	<i>Product Backlog</i>	Pre-Juego
RAPE	A1.5 Identificar y documentar	<i>Scrum</i>	<i>Product Backlog</i>	Pre-Juego

Rol Moprosoft	Descripción Moprosoft	Rol Scrum	Artefactos Scrum	Fase de Scrum
	la relación y dependencia de cada una de las actividades.	<i>Master</i> Equipo		
RAPE RDM	A1.6. Establecer el Tiempo Estimado para desarrollar cada actividad.	<i>Scrum</i> <i>Master</i> Equipo	<i>Product Backlog</i>	Pre-Juego
RAPE	A1.7. Elaborar el Plan de Adquisiciones y Capacitación, definiendo las características y el calendario en cuanto a recursos humanos, materiales, equipo y herramientas. Incluir la capacitación requerida para que el equipo de trabajo pueda desempeñar el proyecto.	<i>Scrum</i> <i>Master</i>		Pre-Juego
RGPY RAPE	A1.8. Conformar el “Equipo de Trabajo”, asignando roles y responsabilidades y basándose en la “Descripción del Proyecto.”	<i>Scrum</i> <i>Master</i> Equipo	<i>Release Planning</i>	Pre-Juego
RAPE	A1.9. Asignar fechas de inicio y fin a cada una de las actividades, a fin de generar el “Calendario de trabajo,” tomando en cuenta los recursos asignados, la secuencia y dependencia de las actividades.	Equipo	<i>Planning Meeting</i>	Pre-Juego
RAPE	A1.10. Evaluar y documentar el “Costo Estimado del proyecto.”	<i>Scrum</i> <i>Master</i>	<i>Planning Meeting</i>	Pre-Juego

**Tabla B.2. Comparación de Actividades entre Moprosoft y Scrum:
fase de planificación (Continuación)**

Rol Moprosoft	Actividades asociadas al rol en Moprosoft	Rol Scrum	Artefactos Scrum	Fase de Scrum
<i>Planificación</i>				
RGPY RAPE RDM	A1.11. Identificar, describir y evaluar los riesgos que pueden afectar el proyecto relacionados con el equipo de trabajo, incluyendo al Cliente y a los usuarios. También incluye riesgos con la tecnología o la metodología, riesgos con la organización del proyecto (costo, tiempo, alcance y recursos) o riesgos externos al proyecto. Identificar la probabilidad e impacto de cada riesgo, estimando sus implicaciones en los objetivos del proyecto (análisis cuantitativo). Priorizar los efectos de los riesgos sobre los objetivos del proyecto (análisis cualitativo). Desarrollar procedimientos para reducir el impacto de los riesgos. Documentar en el “Plan de Manejo de Riesgos” o actualizarlo.	Equipo	<i>Product Backlog/List of impediments</i>	Pre-Juego/ <i>Sprint</i>
RAPE	A1.12. Generar el “Plan del Proyecto” o actualizarlo antes de iniciar un nuevo ciclo.	Equipo	<i>Product Backlog</i>	Pre-Juego
RAPE	A1.13. Generar el “Plan de Desarrollo” en función del “Plan del Proyecto” o actualizarlo antes de iniciar un nuevo ciclo.	Equipo	<i>Product Backlog</i>	Pre-Juego

**Tabla B.3. Comparación de Actividades entre Moprosoft y Scrum:
fase de Realización**

Rol Moprosoft	Actividades asociadas al rol en Moprosoft	Rol Scrum	Artefactos Scrum	Fase de Scrum
Realización				
RAPE RDM	A2.1. Acordar con el Responsable de “Desarrollo y Mantenimiento del proyecto” la asignación de tareas al “Equipo de Trabajo” incluyendo a los subcontratistas.	<i>Scrum Master</i>	<i>Product Backlog</i>	Pre-Juego

**Tabla B.4. Comparación de Actividades entre Moprosoft y Scrum:
fase de Cierre.**

Rol Moprosoft	Actividades asociadas al rol en Moprosoft	Rol Scrum	Artefactos Scrum	Fase de Scrum
Cierre				
RAPE CL	A4.1. Formalizar la terminación del ciclo o del proyecto de acuerdo al Protocolo de Entrega establecido en el “Plan del Proyecto” y obtener el “Documento de Aceptación.”	Equipo	Condiciones de Hecho (<i>do done</i>) <i>Spring Planning</i>	Pre-Juego
RDM	A2.10. Documentar la versión preliminar del “Manual de Usuario” o modificar el manual existente.	No soportado por <i>Scrum</i>		
RDM	A2.13. Incorporar la “Especificación de Requerimientos” y el “Manual de Usuario” a la “Configuración de Software.”	No soportado por <i>Scrum</i>		

Nota. La fase de evaluación y control no se incorporan en las tablas debido a que las actividades son propias de los niveles de gestionado y establecido la única actividad para el nivel de estudio es la realización de la documentación de aceptación, la cual dentro del modelo ágil no es soportado por Scrum.

Anexo C. Comparación de Roles Moprosoft y *Scrum* del proceso de “Desarrollo y Mantenimiento de Software”

En este anexo se presenta un análisis de comparación entre roles y actividades definidos en Moprosoft y *Scrum* para el proceso de “Desarrollo y Mantenimiento de Software”. La tabla C.1 compara roles involucrados; la tabla C.2 compara actividades de la fase de inicio; la tabla C.3 compara actividades de requerimientos; la tabla C.4 compara actividades de realización de análisis y diseño; la tabla C.5 compara actividades de Construcción.

Tabla C.1. Comparación de Roles involucrados en el proceso de Moprosoft y *Scrum*: “Desarrollo y Mantenimiento de Software”

Rol de Scrum	Rol de Moprosoft
<i>Scrum Master</i>	Responsable del proceso “Administración del Proyecto Específico” (RAPE)
<i>Scrum Master</i>	Responsable del proceso “Desarrollo y Mantenimiento de Software” (RDM)
Integrante de equipo	Analista (AN)
Integrante de equipo	Diseñador de Interfaz de Usuario (DU)
Integrante de equipo	Diseñador (DI)
Integrante de equipo	Programador (PR)
Integrante de equipo	Responsable de Manuales (RM)
Equipo	Equipo de Trabajo (ET)
<i>Product Owner</i>	Cliente (CL)
Cliente	Usuario (US)

Tabla C.2. Comparación de Actividades entre Moprosoft y *Scrum*: Realización de la fase de Inicio.

Rol Moprosoft¹⁰	Actividades asociadas al rol en Moprosoft	Rol Scrum	Artefactos Scrum	Fase Scrum
<i>Realización de la fase de inicio</i>				
ET	A1.1. Revisar con los miembros del equipo de trabajo el “Plan de Desarrollo” actual para lograr un entendimiento común y obtener su compromiso con el proyecto.	Equipo	<i>Sprint Backlog</i>	Pre-Juego

¹⁰ Ver tabla C.1 para una descripción de las iniciales de los roles

**Tabla C.3. Comparación de Actividades entre Moprosoft y Scrum:
Realización de la fase de Requerimientos.**

Rol Moprosoft	Actividades asociadas al rol en Moprosoft	Rol Scrum	Artefactos Scrum	Fase Scrum
Realización de la fase de requerimientos				
RDM AN	A2.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al “Plan de Desarrollo” actual.	<i>Scrum Master</i> Equipo	<i>Sprint Backlog</i> <i>Scrum Taskboard</i>	Pre-Juego/ <i>Daily Scrum Meeting</i>
AN CL US DU	A2.2. Documentar o modificar la “Especificación de Requerimientos.” Identificar y consultar fuentes de información (clientes, usuarios, sistemas previos, documentos, etc.) para obtener nuevos requerimientos. Analizar los requerimientos identificados para delimitar el alcance y su factibilidad, considerando las restricciones del ambiente del negocio del cliente o del proyecto. Elaborar o modificar el prototipo de la interfaz con el usuario. Generar o actualizar la Especificación de requerimientos. Las actividades A2.3 a la A2.9 son de Validación y Verificación.	<i>Scrum Master</i> Equipo Equipo	Historias de Usuario en cada iteración <i>Sprint Backlog</i> <i>Sprint Backlog</i> <i>Sprint Backlog</i> <i>Sprint Backlog</i> <i>TDD</i>	Pre-Juego/ <i>Daily Scrum Meeting</i> <i>Sprint</i>
RM	A2.10. Documentar la versión preliminar del “Manual de Usuario” o modificar el manual existente.	No soportado por <i>Scrum</i>		
RDM	A2.13. Incorporar la “Especificación de Requerimientos” y el “Manual de Usuario” a la Configuración de Software.	No soportado por <i>Scrum</i>		

**Tabla C.4. Comparación de Actividades entre Moprosoft y Scrum:
Realización de la fase de Análisis y Diseño.**

Rol Moprosoft	Actividades asociadas al rol en Moprosoft	Rol Scrum	Artefactos Scrum	Fase Scrum
Realización de la fase de Análisis y Diseño				
RDM AN DI	A3.1 Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al “Plan de Desarrollo” actual.	<i>Scrum Master</i> Equipo	<i>Sprint Backlog</i>	Pre-Juego/ <i>Daily Scrum Meeting</i>
AN DI DU	A3.2. Documentar o modificar el Análisis y Diseño. Analizar la “Especificación de Requerimientos” para generar la descripción de la estructura interna del sistema y su descomposición en subsistemas y éstos a su vez en componentes, definiendo las interfaces entre ellos. Describir el detalle de la apariencia y el comportamiento de la interfaz con base en la “Especificación de Requerimientos” de forma que se puedan prever los recursos para su implementación. Describir el detalle de los componentes que permita su construcción de manera evidente. Generar o actualizar el documento de “Análisis y Diseño.”	<i>Scrum Master</i> Equipo		<i>Daily Scrum Meeting</i>
RDM	A3.10. Incorporar el documento “Análisis y Diseño” a la “Configuración de Software.”	<i>Scrum Master</i> Equipo	<i>Test Driver Development</i>	<i>Daily Scrum Meeting</i>

Nota. La fase de integración y pruebas no se incorporan en las tablas debido a que las actividades son propias de los niveles de gestionado, establecido y predecible la única actividad, las actividades del nivel 1 de esta fase no es soportado por *Scrum*.

Tabla C.5. Comparación de Actividades entre Moprosoft y Scrum:
Realización de la fase de Construcción.

Rol Moprosoft	Actividades asociadas al rol en Moprosoft	Rol Scrum	Artefactos Scrum	Fase Scrum
Realización de la fase de Construcción				
RDM	A4.1. Distribuir tareas a los miembros del equipo de trabajo según su rol, de acuerdo al “Plan de Desarrollo” actual.	Equipo	Product Backlog Scrum Taskboard	Pre-Juego/Daily Scrum Meeting
PR	A4.2. Construir o modificar el(los) componente(s) de software con base en la parte detallada del “Análisis y Diseño.”	Equipo	Test Driver Development	Daily Scrum Meeting
RDM	A4.5. Incorporar los componentes a la configuración de Software.	Equipo		Sprint
PR RPU	A5.2. Realizar integración de los componentes en subsistemas o en el sistema del Software	Equipo		Sprint
RM	A5.3. Documentar el “Manual de Operación” o modificar el manual existente.	No procede		
RM	A5.8. Documentar el “Manual de Usuario” o modificar el existente.			
RDM	A5.11. Incorporar “Software,” “Manual de Operación” y “Manual de Usuario” a la “Configuración de Software.”			

Anexo D: Descripción Detallada de Competencias

En este anexo se presenta un análisis de comparación entre roles y competencias definidos en Moprosoft, y Scrum para los proceso de “Administración de Proyectos y Desarrollo y Mantenimiento de Software”. El Tabla D.1 se presentan las competencias para el proceso de “Administración de proyectos específicos” y el D.2 presenta las correspondientes al “Desarrollo y mantenimiento de software.”

Tabla D.1. Competencias para cubrir el proceso de Administración de Proyectos Específicos en CAAM

Realiza	Entrada a CAAM	Competencias de la capacitación. Ser capaz de:	Salida de CAAM	Actividades cubiertas en Moprosoft
<i>Master Scrum, Cliente</i>	Descripción del proyecto. Definición de entrega de cada uno de los entregables. Reuniones con el cliente	C1. Identificar cada uno de los elementos que debe de comprender la planeación de un desarrollo de software en un ambiente ágil. C2. Diseñar estimaciones para diversos elementos como el tiempo y esfuerzo para el equipo de trabajo ágil.	<i>Product Backlog</i>	Planificación: A1.1, A1.4, A1.5, A1.6, A1.7
<i>Master Scrum Cliente</i>	<i>Product Backlog</i> “Historias de usuario.” “Información histórica.” “Metas cuantitativas.”	C3. Planear en base a las historias de usuario, los tiempos de inicio y fin de cada una de las actividades en un desarrollo ágil.	<i>Planning Meeting</i>	Planificación: A1.8, A1.9 A1.12, A1.13
Master Scrum	Descripción del proyecto	C4. Identificar la probabilidad e impacto de cada riesgo en el desarrollo ágil. C5. Desarrollar procedimientos para la reducción de riesgos en desarrollos ágiles.	Plan de Manejo de Riesgos	Planificación: A1.11
Equipo	Descripción del proyecto <i>Product Backlog</i>	C6. Diseñar una tabla de compromisos para ser logrados en cada iteración del desarrollo ágil.	<i>Sprint Backlog</i>	Fase de Inicio: A1.1

**Formato D.2. Competencias para cubrir el proceso de Desarrollo
y mantenimiento de software en CAAM**

Realiza	Entrada	Competencias de la capacitación. Ser capaz de:	Salida	Actividades cubiertas en Moprosoft
Cliente, Equipo	Lista de requisitos entregada por el cliente.	C7. Desarrollar un plan de iteraciones que le permita conseguir los resultados deseados por el cliente.	“Plan de Iteraciones”	Fase de análisis y diseño: A3.1 Realización de la fase de construcción: A4.1.
<i>Scrum Master, Equipo</i>	Lista de “Historias de Usuario”	C8. Organizar una lista con prioridades de historias para desarrollarlas mediante estrategias ágiles.	<i>Scrum Taskboard</i>	Fase de Análisis y Diseño: A3.2
Equipo	“Historia de Usuario”	C9. Desarrollar un componente mediante la filosofía de desarrollo de TDD bajo la restricción que se indique.	Diseño del componente. Componente verificado mediante prueba de unidad	Realización de la construcción: A4.2, A4.5, A5.2.

Anexo E: Encuesta Inicial de implementación de CAAM.

Este anexo muestra el cuestionario que se aplicará a los miembros del departamento de desarrollo de software cuando se inicie la implementación del modelo de mejora de capacitación CAAM. El objetivo del cuestionario es evaluar la posición actual de los desarrolladores con respecto a su metodología de desarrollo.

ENCUESTA PARA DESARROLLADORES MODELO CAAM CUESTIONARIO TIPO 1

El presente cuestionario tiene como finalidad determinar el grado de las prácticas que actualmente se presentan en su equipo de desarrollo de software, para los conceptos descritos en cada enunciado de la encuesta.

ANTECEDENTES. Favor de seleccionar con una X alguna de las opciones siguientes.

El número de empleados en tu organización es:		
	Menos de 10	
	Entre 10-20	
	Entre 21-40	
	Entre 41-100	
	Superior a 100	

El número de empleados en tu equipo es:		
	Menos de 5	
	De 5 a 10	
	De 11 a 20	
	De 21 a 40	
	Mayor de 40	

Tu papel en el equipo es de:		
	Gerente	
	Administrador(a) de proyectos	
	Jefe(a) de equipo	
	Desarrollador(a)	

	Otros. Por favor, especifica : _____
--	--------------------------------------

El tiempo que has desarrollado software, utilizando los principios de Moprosoft es:		
	Nunca	
	Menos de 1 año	
	De 1-3 años	
	De 3-5 años	
	Más de 5 años	

En la siguiente sección, por favor marca una X en la columna correspondiente, en la escala de 1 (si estás totalmente en desacuerdo) al 5 (si estás totalmente de acuerdo).

	Respuestas					
	Totalmente en desacuerdo 1	Algo en desacuerdo 2	Ni en desacuerdo ni en acuerdo 3	Algo de acuerdo 4	Totalmente de acuerdo 5	No es aplicable o no sé
1. En mi equipo se da prioridad a la satisfacción de los clientes, a través de la entrega oportuna.						
2. En mi equipo se entrega software funcional de manera frecuente, en un periodo de dos semanas a dos meses.						
3. En mi equipo se aceptan cambios a los requerimientos durante el desarrollo.						
4. En mi equipo el trabajo de construcción de software se realiza de una manera conjunta con el desarrollador y la gente de negocios.						
5. Mi equipo desarrolla su trabajo en un medio ambiente adecuado.						
6. En mi equipo se hace hincapié en la						

	Respuestas					
	Totalmente en desacuerdo 1	Algo en desacuerdo 2	Ni en desacuerdo ni en acuerdo 3	Algo de acuerdo 4	Totalmente de acuerdo 5	No es aplicable o no sé
comunicación cara a cara entre los usuarios y desarrolladores.						
7. En mi equipo podemos medir el progreso del desarrollo de software en cualquier momento.						
8. Mi equipo de desarrollo es auto-organizado.						
9. Mi equipo puede lograr cumplir con las necesidades cambiantes y los nuevos retos en cada proyecto.						
10. En mi equipo hacemos uso de modelos simples para el desarrollo de software.						
11. En mi equipo se cuenta con un mecanismo de pruebas efectivo que asegure lo que realmente requiere el usuario.						

Anexo F: Encuesta Final de implementación de CAAM.

El cuestionario final aplicado a los usuarios de CAAM tiene como objetivo determinar el grado de uso de las prácticas adoptadas cada vez que se avanza en la capacitación a través del uso del modelo CAAM. Se espera que este cuestionario sea aplicado a los mismos integrantes de equipos desarrolladores de software, a los que aplicó el cuestionario inicial descrito en el anexo E. La tabla F.1 describe las habilidades obtenidas en cada pregunta del cuestionario.

ENCUESTA PARA DESARROLLADORES MODELO CAAM CUESTIONARIO TIPO 2

El presente cuestionario tiene como finalidad determinar el grado de las prácticas que actualmente se presentan en su equipo de desarrollo de software, para los conceptos descritos en cada enunciado de la encuesta.

	Respuestas					
	Totalmente en desacuerdo	Algo en desacuerdo	Ni en desacuerdo ni acuerdo	Algo de Acuerdo	Totalmente de acuerdo	No es aplicable o no sé
	1	2	3	4	5	
1. En mi equipo se adopta como prioridad alta el lograr la satisfacción del cliente.						
2. En mi equipo se adopta la colaboración estrecha con los clientes para el desarrollo del proyecto.						
3. En mi equipo se identifica que la colaboración en los equipos pequeños existe una comunicación eficaz.						

	Respuestas					
	Totalmente en desacuerdo	Algo en desacuerdo	Ni en desacuerdo ni acuerdo	Algo de Acuerdo	Totalmente de acuerdo	No es aplicable o no sé
	1	2	3	4	5	
4. Existen respuestas rápidas a los cambios y se llevan mejores tomas de decisiones en el desarrollo de software.						
5. En mi equipo se llevan mejores decisiones en la construcción del software comparadas con las tomadas antes de conocer a CAAM.						
6. Comparado con hace 2 meses la capacidad de entrega temprana y continua de mi equipo ha mejorado.						
7. Comparado con hace 2 meses mi equipo ha mejorado la verificación temprana de posibles riesgos en la entrega de productos en cada iteración.						
8. Mi equipo adopta cómo método más productivo la comunicación entre sus miembros “cara a cara.”						
9. En mi equipo se adopto como filosofía “que las decisiones de un						

	Respuestas					
	Totalmente en desacuerdo	Algo en desacuerdo	Ni en desacuerdo ni acuerdo	Algo de Acuerdo	Totalment e de acuerdo	No es aplicable o no sé
	1	2	3	4	5	
proyecto se tomen en tiempos cortos y con rapidez.”						
10. El tiempo de desarrollo del último producto en el que se involucró mi equipo fue controlado de alguna manera.						
11. En mi equipo se ha tomado la habilidad de planear la entrega del tiempo y forma.						

Tabla F.1 Identificación de competencias logradas que deben de repercutir en la actitud del desarrollador.

Competencias de la capacitación. Ser capaz de:	Preguntas
Identificar cada uno de los elementos que debe de comprender la planeación de un desarrollo de software en un ambiente ágil.	6, 11
Diseñar estimaciones para diversos elementos como el tiempo y esfuerzo para el equipo de trabajo ágil mediante restricciones del usuario.	6, 8, 10
Planear en base a las historias de usuario, los tiempos de inicio y fin de cada una de las actividades en un desarrollo ágil.	7, 8, 9
Identificar la probabilidad e impacto de cada riesgo en el desarrollo ágil.	3
Desarrollar procedimientos para la reducción de riesgos en desarrollos ágiles.	5
Diseñar una tabla de compromisos para ser logrados en cada iteración del desarrollo ágil.	1, 2
Desarrollar un plan de iteraciones que le permita conseguir los resultados deseados por el cliente.	1, 2
Organizar una lista con prioridades de historias para desarrollarlas mediante estrategias ágiles.	3, 4
Desarrollar un componente mediante la filosofía de desarrollo de TDD bajo la restricción que se indique.	7

Anexo G: Curso de validación de hipótesis del proceso de CAAM.

La siguiente planeación identifica las actividades a realizar teniendo en cuenta la competencia a lograr, la cual permitirá dar cumplimiento al cambio de nivel del 0 al 1 en Moprosoft.

Universidad Popular Autónoma del Estado de Puebla
Planeación de Actividades para la capacitación de Scrum para empresas con moprosoft 1
Tesis: Modelo de Mejora del Proceso de capacitación para desarrolladores de software
en PYMES basado en ambientes ágiles

Definiciones/Artefatos	Temas	Horas								Competencia.
		Teóricas					Prácticas			
		1	2	3	4	5	6	7	8	
Definición: Roles Definición: Artefactos Definición: Herramientas Definición: Ciclo de vida		■								
Creación de la lista de productos	Preparación de Historias de Usuario		■				■			C1. Identificar cada uno de los elementos que debe de comprender la planeación de un desarrollo de software en un ambiente ágil.
	Estimación inicial de la historia de usuario			■				■		C2. Diseñar estimaciones para diversos elementos como el tiempo y esfuerzo para el equipo de trabajo ágil mediante restricciones del usuario.
	Planificación de una iteración.				■				■	C3. Planear en base a las historias de usuario, los tiempos de inicio y fin de cada una de las actividades en un desarrollo ágil.
	Cómo probar la Historia de Usuario					■				C4. Identificar la probabilidad e impacto de cada riesgo en el desarrollo ágil.
Reunión de planificación de la iteración	Selección de Historias según estimación					■				C5. Desarrollar procedimientos para la reducción de riesgos en desarrollos ágiles.
	Descomposición de la historia en tareas						■			C6. Diseñar una tabla de compromisos para ser logrados en cada iteración del desarrollo ágil.
	Seguimiento y control ágil							■		C7. Desarrollar un plan de iteraciones que le permita conseguir los resultados deseados por el cliente.
Hacer pruebas	Probar Primero								■	C8. Organizar una lista con prioridades de historias para desarrollarlas mediante estrategias ágiles.

