DEFINICIÓN DE ESTRATEGIAS PARA LA GESTIÓN DE CONOCIMIENTO EN PROYECTOS QUE USAN METODOLOGÍAS AGILES DE DESARROLLO DE SOFTWARE: CASO MVM INGENIERÍA DE SOFTWARE S.A.S

Zuleima Marcela Moreno Ruiz

ESPECIALIZACIÓN EN INGENIERÍA DE SOFTWARE
TRABAJO DE GRADO

ASESORES
Temático: MSc. Ricardo Alonso Gallego Burgos
Metodológico: MSc. Bell Manrique Losada

UNIVERSIDAD DE MEDELLIN
FACULTAD DE INGENIERIA
MEDELLIN
2014

TABLA DE CONTENIDO

LISTADO DE FIGURAS

Capítulo 1.

Capítulo 2.

Capítulo 3.

Capítulo 4.

LISTADO DE TABLAS

Capítulo 1

Capítulo 3

Capítulo 4

INTRODUCCION

PLANTEAMIENTO DEL PROBLEMA

En los últimos años, el conocimiento se ha convertido en un factor primordial para las organizaciones, su gestión les permite ofrecer productos y servicios más competitivos y con alto componente innovador, su aparición y creciente importancia ha generado que estas se enfoquen en el desarrollo de tecnologías, metodologías y estrategias que les permitan su medición, creación y difusión (Rodríguez, 2006; *Crawford et al.*, 2006; *Chau* et al., 2003; *Kavitha* y *Irfan*, 2011). Por lo anterior, MVM Ingeniería de Software S.A.S, ha entendido la importancia de definir mecanismos para la adopción y transferencia de conocimiento en los procesos de ingeniería, sus esfuerzos se han enfocado en la gestión de la documentación del ciclo de vida de las metodologías tradicionales de desarrollo de software. Es así como su sistema de gestión del conocimiento, esta soportado en el modelo de conversión de conocimiento propuesto por *Nonaka* y *Takeuchi* (2000). En la actualidad la organización ha iniciado la implementación de metodologías agiles en sus procesos de desarrollo de software, en donde la transferencia de conocimiento se soporta en las redes conversacionales, la mayoría de la documentación generada es reemplazada por la comunicación cara a cara entre los miembros del proyecto y el cliente, por lo que se hace necesaria la definición de estrategias que permitan la adopción y transferencia de conocimiento, bajo este tipo de metodologías, que le permitan a MVM mantener una dinámica de aprendizaje permanente que conlleve a la generación de ventaja competitiva, y se disminuyan los riesgos inherentes al crecimiento de la fábrica de software, y la rotación de sus colaboradores.

Por esta razón, se plantea la pregunta:
¿Qué estrategias se pueden proponer para fortalecer los procesos de adopción y transferencia de conocimiento en los proyectos que utilizan metodologías ágiles para el desarrollo de software en MVM Ingeniería de Software S.A.S?

OBJETIVO GENERAL

Definir estrategias que permitan fortalecer la adopción y transferencia de conocimiento en los proyectos que utilizan metodologías agiles de desarrollo de software en MVM Ingeniería de Software S.A.S.

OBJETIVOS ESPECIFICOS

- Examinar la literatura existente para identificar los modelos de gestión del conocimiento que se han adoptado en los proyectos de desarrollo de software que usan metodologías agiles.

- Describir la forma en que MVM Ingeniería de Software utiliza metodologías agiles en el proceso de desarrollo de software.

- Identificar los principales componentes del sistema de gestión del conocimiento que tiene implementado MVM Ingeniería de Software S.A.S., y como éstos pueden ser incorporados en proyectos que usan metodologías agiles.

- Definir estrategias para fortalecer las actividades de adopción y transferencia de conocimiento en los proyectos que llevan a cabo metodologías ágiles en MVM Ingeniería de Software S.A.S.

JUSTIFICACION

La gestión del conocimiento simplifica el proceso de compartir, distribuir, crear, capturar y comprender el conocimiento de una compañía (*Olav* y *Dingsøyr*, 2008), facilita la búsqueda, codificación, sistematización y difusión de las experiencias individuales y colectivas del talento humano de la organización y se ha convertido en un factor

primordial que permite que la organización pueda ofrecer productos y servicios competitivos (Farfán y Garzón, 2006).

Por lo anterior, MVM Ingeniería de Software S.A.S ha entendido la importancia de definir mecanismos y estrategias para la adopción y transferencia de conocimiento, en los procesos de desarrollo de software que implementan metodologías ágiles, en donde la transferencia de conocimiento se soporta en las redes conversacionales y la mayoría de la documentación es reemplazada por la comunicación cara a cara entre los miembros del equipo y el cliente. Lo anterior le permitirá a MVM mantener una dinámica de aprendizaje permanente que conlleve a la generación de ventaja competitiva y a la mitigación de riesgos inherente al crecimiento de la fábrica de software y rotación de los colaboradores.

METODOLOGIA

La estrategia de investigación a aplicar en el presente trabajo, es una metodología cualitativa, de tipo exploratorio y con un método de estudio de caso, donde la unidad de análisis es la empresa MVM Ingeniería de Software S.A.S. Esta metodología permitirá diseñar estrategias para fortalecer las actividades de adopción y transferencia de conocimiento de los proyectos que utilizan metodologías agiles de desarrollo de software en la empresa, a partir de una serie de suposiciones extraídas del marco conceptual que se elabore en esta investigación. La razón por la cual este trabajo se adopta un tipo de investigación cualitativa, está dada por la necesidad de obtener información a partir de alguna ya existente, que sirve como apoyo al investigador, para contrarrestar la realidad, sin tener la necesidad de extraer una muestra real del fenómeno, tal cual sucede en investigaciones de tipo cuantitativo. El propósito del método de estudio de caso, será exploratorio, a través del cual se pretende identificar los principales elementos de la gestión del conocimiento tácito generado en la aplicación de metodologías agiles en el desarrollo de software y como pueden ser incorporados en el diseño de las estrategias para la adopción y transferencia de conocimiento en MVM. El estudio de caso, se apoya en la metodología planteada por diversos autores.

De acuerdo con Yin (1989), el método de estudio de caso en las investigaciones cualitativas, es útil para temáticas que se consideran nuevas y en las cuales se tiene los siguientes rasgos distintivos: examina o indaga sobre un fenómeno contemporáneo en su entorno real, la frontera entre el fenómeno y su contexto no son claramente evidentes, se utilizan diversas fuentes de datos y puede estudiarse un caso único o múltiple. Por otra parte, *Chetty* (1996), sostiene que el método de estudio de caso es una metodología rigurosa, que se permite: adecuada para investigar y dar respuesta al cómo y por qué ocurren los fenómenos, permite estudiar un tema determinado, permite estudiar los fenómenos desde múltiples perspectivas y no desde la influencia de una variable, permite explorar de forma más profunda y obtener un conocimiento más amplio, sobre cada fenómeno. La metodología cualitativa, permite explicar de nuevos fenómenos y en la elaboración de teorías en las que los elementos de carácter intangible, tácito o dinámico juegan un papel determinante. Además, el estudio de caso es capaz de satisfacer todos los objetivos de una investigación y también analizar diferentes casos con distintas intenciones (Sarabia, 1999).

Título del trabajo: DEFINICIÓN DE ESTRATEGIAS PARA LA GESTIÓN DE CONOCIMIENTO EN PROYECTOS QUE USAN METODOLOGÍAS AGILES DE DESARROLLO DE SOFTWARE: CASO MVM INGENIERÍA DE SOFTWARE S.A.S

Autor: Zuleima Marcela Moreno Ruiz
Título otorgado: Especialista en Ingeniería de Software
Asesor del trabajo: MSc. Ricardo Alonso Gallego Burgos

Programa de donde egresa: Especialización en Ingeniería de Software
Ciudad: Medellín
Año: 2014

RESUMEN

En la actualidad, la implementación de tecnologías, metodologías y estrategias que permitan la adopción, transferencia y mejoramiento continuo del conocimiento, se ha convertido en un factor importante para que las organizaciones aumenten su ventaja competitiva (Rodríguez Gómez, 2006). En este sentido las empresas de desarrollo de software que han adoptado metodologías agiles, en donde la transferencia de conocimiento se soporta en las redes conversacionales, requieren mecanismos de transferencia de conocimiento tácito y explicito, por lo cual, este trabajo de investigación, está orientado a definir un conjunto de estrategias que permitan garantizar la gestión del conocimiento en proyectos que implementan este tipo de metodologías para el desarrollo de software, por medio del uso del método de estudio de caso en la organización MVM Ingeniería de Software S.A.S.

1. MARCO TEÓRICO

El conocimiento se ha convertido en un factor de gran importancia para las organizaciones, principalmente aquellas que producen y/o mantienen software, lo que ha generado que se enfoquen en definir actividades que le permitan crear, transferir y difundir el conocimiento para aumentar su ventaja competitiva (Rodríguez, 2006). En este capítulo se describen, en forma general, la información teórica analizada en la investigación, donde se abordan los principales conceptos acerca del conocimiento y su gestión), caracterización y definición de desarrollo ágil de software, y una visión de la investigación a través del resumen de los estudios existentes de la relación de la gestión del conocimiento con las metodologías ágiles de desarrollo de software.

1.1. GESTION DEL CONOCIMIENTO

1.1.1. CONOCIMIENTO

Según *Probst* et al. (2001), el conocimiento es todo el conjunto de cogniciones y habilidades con el cual los individuos suelen solucionar problemas. Se basa en datos e información, pero a diferencia de éstos siempre está ligado a las personas, forma parte integral de los individuos y representa las creencias de éstos acerca de las relaciones causales (Pereira, 2011).

Nonaka et al. (2000) conciben al conocimiento como un activo creado y mantenido de forma colectiva a través de la interacción entre individuos o entre individuos y su entorno, más que creado de forma individual. Para expresar esta idea, dichos autores se refieren al "Ba", entendido como el contexto compartido en el que el conocimiento es creado, compartido y utilizado.

Tsoukas y *Vladimirou* (2001), señalan que el conocimiento es la capacidad individual para realizar distinciones o juicios en relación a un contexto, teoría o ambos; respecto el conocimiento organizacional es cuando los individuos son capaces de realizar distinciones sobre el contexto en el que actúan y, además obedecen a un conjunto de reglas genéricas producidas por la organización (*Segarra* y *Bou*, 2004).

En tecnología de la información, el conocimiento se diferencia de los datos y la información, los datos son una colección de hechos, medidas y estadísticas. Según *Husemann* y *Goodman* (1999), los datos son hechos objetivos que describen un evento sin ningún juicio, perspectiva o contexto, cuando se analizan los datos para añadir comprensión, pertinencia, sentido y propósito, se crea información. *Drucker* (1988) define la información como datos mejorados con relevancia y propósito. El conocimiento es entonces la transformación y enriquecimiento de la información por la experiencia personal, las creencias y los valores que aportan a la toma de decisiones (*Nidhra*, 2013).

La definición para este trabajo se basa en la propuesta por Davenport y *Prusak* (1998), donde el conocimiento es un flujo en el que se mezclan la experiencia, los valores importantes, la información contextual y los puntos de vista de expertos, que facilitan un marco de análisis para la evaluación e incorporación de nuevas experiencias e información. Se origina y es aplicado en la mente de los conocedores (*Segarra* y Bou, 2004).

1.1.2. TIPOS DE CONOCIMIENTO

En la literatura, existen diferentes clasificaciones para el conocimiento. *Nonaka* (1995) distingue entre conocimiento implícito (tácito) y conocimiento explicito, el primero se refiere al conocimiento que se almacena en los libros de texto, productos de software y documentos, y el segundo es aquel conocimiento que se almacena en la mente de las personas, en forma de memoria, de habilidades, de experiencia, de educación, de imaginación y de creatividad (*Levy* y *Hazzan*, 2009). *Alavi* y *Leidner* (2001), Baker et al. (1997), *Civi* (2000), *Gupta et al.* (2000), *Lee* y *Yang* (2000), y *Martensson* (2000), han

generado diversas discusiones sobre la clasificación del conocimiento, donde definen el conocimiento tácito como aquel que reside principalmente en la mente de las personas y es relativamente difícil de expresar; y el conocimiento explicito como aquel conocimiento que se ha articulado, codificado y formalizado en alguna forma electrónica o física (*Wong* y *Aspinwall*, 2006).

Por su parte, *Lalangui* (2008) define el ***Conocimiento Tácito***, como aquel tipo de conocimiento que permanece en un nivel "Inconsciente", se encuentra desarticulado y lo implementamos y ejecutamos de una manera mecánica sin darnos cuenta de su contenido. El ***Conocimiento Implícito*** lo define como aquel conocimiento que sabemos que tenemos, pero no nos damos cuenta que lo estamos utilizando, simplemente lo ejecutamos y ponemos en práctica de una manera habitual. Y el ***Conocimiento Explícito*** como aquel tipo de conocimiento que sabemos que tenemos y somos plenamente conscientes cuando lo ejecutamos, es el más fácil de compartir con los demás ya que se encuentra estructurado y muchas veces esquematizado para facilitar su difusión.

Crawford et al., (2006) clasifican el conocimiento como Conocimiento Centrado en el Producto y Conocimiento Centrado en el Proceso, estos enfoques adoptan diferentes perspectivas en relación con la documentación y las interacciones entre las partes interesadas; el conocimiento centrado en el producto se basa en la documentación del conocimiento (creación de documentos, repositorios y posibilidad de reutilización), considera que el conocimiento es una “cosa” que puede ser manipulada como un objeto independiente, es posible capturar, distribuir, medir y gestionar, y el conocimiento centrado en el proceso, es un proceso de comunicación y colaboración, donde el conocimiento está estrechamente ligado con la persona que lo desarrolla y es compartido con otros a través del contacto persona a persona, Pone énfasis en las formas de promover, motivar, estimular o guiar el proceso de aprendizaje, descartando la idea de tratar de capturar y distribuir conocimiento, esta visión entiende principalmente KM como un proceso de comunicación social, que puede ser mejorado por las herramientas de apoyo a la colaboración y la cooperación.

En la Tabla 1-1 se presenta un resumen de las principales clasificaciones del conocimiento realizadas por diversos autores:

Tabla 1-1. Clasificación Tipos de Conocimiento (*Segarra* y *Bou*, 2004)

Estudios	Tipos de Conocimiento
Blackler (1995)	• Conocimiento Cerebral (*Embrained*) • Conocimiento Corporal (*Embodied*) • Conocimiento Incorporado en la Cultura (*Encultured*) • Conocimiento Incrustado en las rutinas (*Embedded*) • Conocimiento Codificado (*Encoded*)
Nonaka y *Takeuchi* (1995)	• Conocimiento Armonizado (de Tácito a Tácito) • Conocimiento Conceptual (de Tácito a Explicito) • Conocimiento Operacional (de Explicito a Tácito) • Conocimiento Sistémico (de Explicito a Explicito)
Spender (1996)	• Conocimiento consciente (Explicito e Individual) • Conocimiento Objetivo (Explicito y Social) • Conocimiento Automático (Implícito e Individual) • Conocimiento Colectivo (Implícito y Social)
Teece (1998)	• Conocimiento Tácito\Conocimiento Codificado • Conocimiento Observable\Conocimiento no Observable en su Uso • Conocimiento Positivo\Conocimiento Negativo • Conocimiento Sistémico\Conocimiento Autónomo
Zack (1999)	• Conocimiento Declarativo • Conocimiento de Procedimiento • Conocimiento Casual
De Long y Fakey (2000)	• Conocimiento Humano • Conocimiento Social • Conocimiento Estructurado
Nonaka et al.	• Activos de Conocimiento basados en la Experiencia

(2000)	• Activos de Conocimiento Conceptual • Activos de Conocimiento Sistémico • Activos de Conocimiento basados en las Rutinas
Alavi y *Leidner* (2001)	• Conocimiento Tácito • Conocimiento Explicito • Conocimiento Individual • Conocimiento Social • Conocimiento Declarativo (*Know-About*) • Conocimiento de Procedimiento (*Know-How*) • Conocimiento Causal (*Know-Why*) • Conocimiento Condicional (*Know-When*) • Conocimiento Relacional (*KnowWith*) • Conocimiento Pragmático

A partir de las diferentes definiciones del conocimiento mencionadas anteriormente, *Nonaka* (1994) define dos dimensiones del conocimiento —Tácito - Explícito e Individual - Colectivo—, donde el conocimiento tácito es aquel difícil de comunicar y articular, es muy personal; el conocimiento explicito es aquel que se puede articular en palabras y números y puede ser compartido en forma de datos, fórmulas científicas y especificaciones; el conocimiento individual es aquel que se crea y existe en los individuos; y el conocimiento colectivo es aquel creado por las acciones colectivas de un grupo y se compone de normas culturales que existen como resultado de trabajar juntos. Con base a la combinación de estas dos dimensiones del conocimiento, Lam (2000) propuso cuatro tipos de conocimiento (*Nidhra* et al., 2013):

- Conocimiento *Embrained*: Individual - Explícito (ej: el conocimiento teórico).
- Conocimiento Incorporado: Individual - Tácito (ej: la experiencia práctica).
- Conocimiento Codificado: Colectivo - Explícito (ej.: reglas escritas, procedimientos).
- Conocimiento Embebido: Colectivo - Tácito (ej.: rutinas, normas).

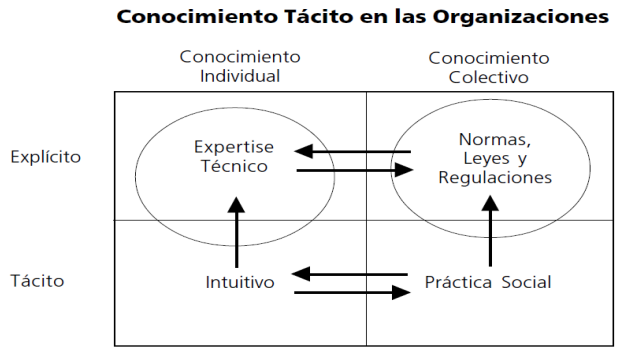


Figura 1-1. Conocimiento Tácito en las Organizaciones (*Nidhra et al*., 2013)

1.1.3. DEFINICION DE LA GESTION DEL CONOCIMIENTO

El conocimiento es considerado como la principal ventaja competitiva de la organización, permitiendo que la empresa sea productiva y pueda ofrecer productos y servicios competitivos, su aparición y creciente importancia, han logrado que las organizaciones aumenten sus esfuerzo para generar tecnologías, metodologías y estrategias que les permitan su medición, creación y difusión (Rodríguez, 2006), con el fin de convertir el conocimiento individual de los miembros de la organización en conocimiento organizacional, logrando que los procesos de la organización no se vean afectados por la rotación del personal, estos esfuerzos por medir, crear y difundir el conocimiento se conocen como “Gestión del Conocimiento”, una disciplina novedosa que no tiene una definición universal, que atraviesa por muchas áreas, y trata una variedad de temas como la economía, la psicología, la informática y la tecnología, por lo que existen varias definiciones según la experiencia del autor y su área de aplicación (Chau et al., 2003). A continuación se enumeran algunas definiciones para poder entender y establecer el significado de este término:

Pereira (2011) presenta las siguientes definiciones para “Gestión del Conocimiento:

- "Proceso sistemático de buscar, organizar, filtrar y presentar la información con el objetivo de mejorar la comprensión de las personas en una área de interés específica" (*Lavenport* y *Klahr*, 1998).

- "Habilidad de desarrollar, mantener, influenciar y renovar los activos intangibles llamados capital de conocimiento o capital intelectual" (*Saint-Ouge*, 1996).

- "Arte de crear valor con los activos intangibles de una organización" (*Sarvary*, 1999).

- "Proceso sistemático de detectar, seleccionar, organizar, filtrar, presentar y usar la información por parte de los participantes de la empresa, con el objeto de explotar cooperativamente el recurso de conocimiento basado en el capital intelectual propio de las organizaciones, orientados a potenciar las competencias organizacionales y la generación de valor" (*Harvard Business Review*, 2003).

Según Nonaka et al. (1999), la Gestión del Conocimiento (GC) es un sistema que facilita la búsqueda, codificación, sistematización y difusión de las experiencias individuales y colectivas del talento humano de la organización, para convertirlas en conocimiento globalizado, de común entendimiento y útil en la realización de todas las actividades de la misma, en la medida que permita generar ventajas sustentables y competitivas en un entorno dinámico (Farfán y Garzón, 2006).Davenport (1998) define la Gestión del Conocimiento como un método para simplificar el proceso de compartir, distribuir, crear, capturar y comprender el conocimiento de una compañía (*Olav* y *Dingsøyr*, 2008).

Gupta et al. (2000) lo define como un proceso que ayuda a las organizaciones a encontrar, seleccionar, organizar, difundir y transferir información y experiencias importantes y necesarias para actividades como la resolución de problemas, el aprendizaje dinámico, la planificación estratégica y la toma de decisiones.

Liebowitz (2003) considera que se trata de capturar, compartir, aplicar y crear conocimiento en una organización, puede entenderse como una aproximación formal y activa para gestionar y optimizar los recursos del conocimiento en una organización (*Wong* y *Aspinwall*, 2006).

Tras un detenido análisis de las definiciones y las características propias de la creación y gestión del conocimiento, podemos considerar que la gestión del conocimiento consiste en un conjunto de procesos sistemáticos orientados al desarrollo organizacional y/o personal y, consecuentemente, a la generación de una ventaja competitiva para la organización y/o el individuo (Rodríguez, 2006)

Para el desarrollo de este trabajo, se define GC como la capacidad de aprender y generar conocimiento nuevo o mejorar el existente, como un sistema facilitador de la búsqueda, codificación, sistematización y difusión de las experiencias individuales y colectivas del talento humano de la organización, para convertirlas en conocimiento globalizado, de común entendimiento y útil en la realización de todas las actividades de la misma, que permita generar ventajas sustentables y competitivas en un entorno dinámico (Farfán y Garzón, 2006). En cierto modo, se puede considerar la gestión del conocimiento como el proceso que intenta entender qué hace el conocimiento, de dónde viene y cómo se crea.

1.1.4. PROCESO DE GESTION DEL CONOCIMIENTO

A partir de la investigación y análisis de las múltiples definiciones y características de la creación y gestión del conocimiento, se puede decir que éste consiste en un conjunto de procesos orientados al desarrollo organizacional y/o personal, y consecuentemente, a la generación de ventaja competitiva para la organización y el individuo. Schneider (2009), define las siguientes, como las tareas para la gestión del conocimiento: los propietarios del conocimiento necesitan ser identificados, y necesitan ser animados o convencidos o forzados a compartir sus conocimientos, el proceso de conversión desde y hacia la forma explícita de conocimiento necesita ser facilitado y soportado por la organización y/o el individuo, se debe garantizar una coincidencia entre piezas de conocimiento que se

necesitan y las que están disponibles, administrar el significado de la información, no es suficiente transferir los datos, también deben haber metadatos que le den significado de los datos en bruto, gestionar y apoyar la intensidad y el estilo de consumo de la información, las experiencias son más que hechos o reglas, también contienen una observación, una emoción y/o una conclusión o hipótesis, si la gestión del conocimiento quiere ofrecer experiencias de pleno derecho, el aspecto emocional tiene que ser capturado, manejado, y entregado, transferir el conocimiento de un punto a otro es importante. A menudo, es necesaria la difusión de una información a muchos receptores. La misma pieza de conocimiento se explota y reutiliza varias veces, multiplicando el beneficio y la validación de la información como un efecto secundario.

Pereira (2010), del Centro de Gestión de Conocimiento CEGESTI, propone que las actividades del ciclo de vida del proceso de gestión del conocimiento se definan así:

1. **Identificar el Conocimiento:** Qué es lo que quiere lograr la empresa y que conocimiento necesita para ello. Se identifica el conocimiento que aún no se ha adquirido (sea tácito o explicito) en todos los niveles (estratégico, funcional, de procesos, personal, etc.). Esta información permite una mejor toma de decisiones y para obtenerla se recomienda utilizar técnicas como los mapas del conocimiento, tormentas de ideas, retroalimentación de los clientes, experiencias en proyectos realizados, etc.

2. **Adquirir el Conocimiento:** Para adquirir el conocimiento es importante tener en cuenta adquirir el conocimiento que se pueda utilizar directamente y aquel que puede ser útil en el futuro. Existen muchos sistemas y métodos para adquirir el conocimiento; por ejemplo, mediante el reclutamiento de consultores especializados, la cacería de talentos, el método de alianzas estratégicas, los convenios de cooperación, por medio del uso de la propiedad intelectual (licenciamiento o franquicias), la ingeniería inversa, el conocimiento documentado (software o CD- ROMS), entre otros.

Figura 1-2. Ciclo de Vida de la Gestión del Conocimiento (Pereira, 2010)

3. **Desarrollar el Conocimiento:** Es importante descubrir cómo se desarrolla el conocimiento en la empresa, teniendo en cuenta los individuos y los grupos, con el fin de considerar el flujo de ideas en procesos de innovación y mejora. Los siguientes son ejemplos de cómo desarrollar esta actividad: programas para el desarrollo de la creatividad para resolver problemas en forma sistemática, procedimientos para la planificación de la innovación y equipos de alto rendimiento que investiguen nuevas tecnologías.

4. **Compartir el Conocimiento:** El objetivo de esta etapa es transferir el conocimiento al lugar donde sea necesario, en el momento adecuado y con la calidad requerida. Por lo tanto, el conocimiento se debe distribuir para que otras personas puedan encontrarlo y lo utilicen en procesos que añadan valor. La mayoría del conocimiento se transmite de mejor forma de persona a persona, mediante la colaboración de talleres, aprendizaje en el trabajo diario, etc. Las

herramientas que facilitan este proceso son la internet, la intranet, *web-conference*, *Skype*, cursos bajo el concepto de *e-learning*, bancos de datos, centros de documentación, rotación del personal, sesiones grupales, reuniones de áreas, teletrabajo, correo electrónico, páginas web, grupos de experiencia, arenas de aprendizaje, etc.

5. **Utilizar el Conocimiento:** El conocimiento añade valor solamente cuando se utiliza en la empresa; el exceso de este no se utiliza a plenitud. El uso del conocimiento determina las necesidades de la empresa a este respecto, y debe servir como referencia para la creación, almacenamiento y las formas de compartir conocimiento.

6. **Retener el conocimiento:** La retención de conocimiento es un paso significativo en la construcción de los activos del conocimiento, puesto que el conocimiento debe incorporarse a la empresa para su reutilización y adquiere mayor relevancia en organizaciones donde existen pocos especialistas en diferentes temas de importancia, y cuya pérdida podría ser negativa para la empresa si se retiran o deciden abandonar la empresa por cualquier motivo. Para que este efecto sea menor, la empresa tiene que ser capaz de almacenar el conocimiento y resguardarlo para que se facilite su transferencia a otros.

Tierney (1999) define dos estrategias diferentes para que las organizaciones adopten en la práctica: codificación y personalización. El primero se refiere al enfoque en el que se extrae el conocimiento de la gente, codificado y capturado en repositorios, de modo que se puede acceder y reutilizar. Esta estrategia también puede ser vista como una manera de retirar el conocimiento de la persona que lo posee, de modo que permanece en una organización. Por el contrario, la personalización se centra en el intercambio de conocimientos a través de persona a persona, los contactos y diálogos. El conocimiento se mantiene dentro de la mente, donde una interacción individual y humana se explota para adquirirla (*Wong* y *Aspinwall*, 2006).

Wong y *Aspinwall* (2006) describen las posibles fuentes de ventajas competitivas de una empresa. Ellos proponen el modelo de cadena de conocimiento que identifica cinco actividades de manipulación de los conocimientos primarios y cuatro actividades secundarias. Las principales actividades son: adquisición, selección, generación, internalización y externalización; las actividades secundarias incluyen: liderazgo, coordinación, control, y gestión del conocimiento.

En resumen podemos indicar que la gestión del conocimiento corresponde a un proceso que implica llevar a cabo las siguientes actividades: identificar, adquirir, desarrollar, compartir, utilizar y retener el conocimiento.

Son varios los autores, que han realizado una descripción del proceso de Gestión del Conocimiento, y han utilizado diferentes términos para referirse a cada una de las etapas del proceso, pero en conclusión, como se pude ver en la Figura 1-3, todas estas propuestas están relacionadas con cuatro fases comunes que abarcan el ciclo de vida de la gestión del conocimiento: (1) Adquisición / Creación / Generación, (2) Retención / Almacenamiento / Captura, (3) Compartir / Transferencia / Difusión, y (4) Aplicación / Utilización / Uso (*Chen* y *Chen*, 2011).

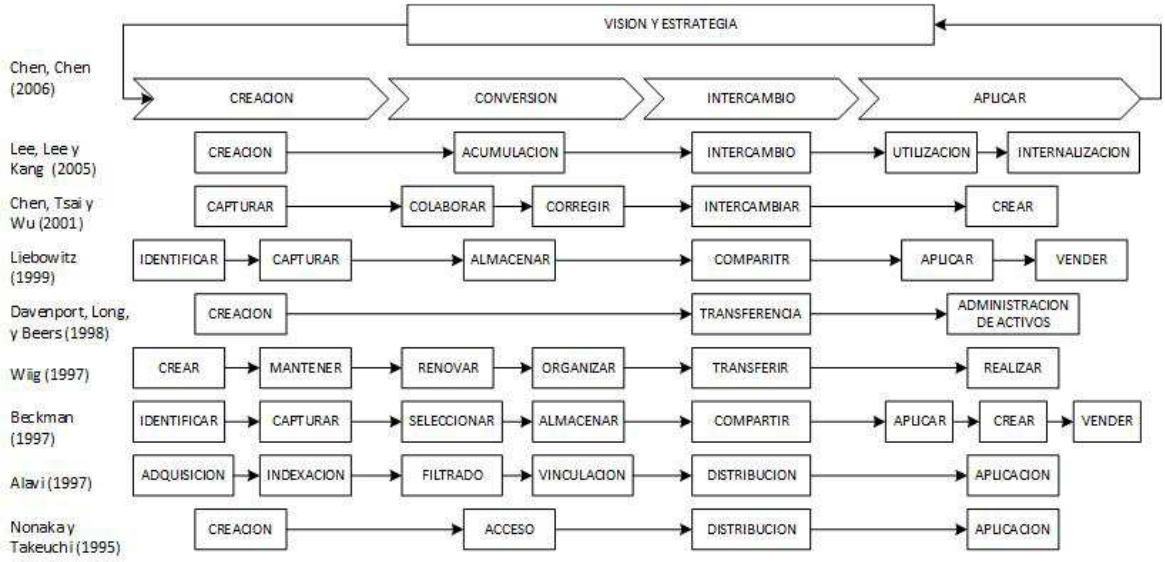


Figura 1-3. Comparativo Gestión del Conocimiento (*Chen* y *Chen*, 2011)

1.1.5. MODELOS DE GESTION DEL CONOCIMIENTO

En la sociedad del conocimiento existen multitud de modelos para la creación y gestión del conocimiento, así como diversas y variadas perspectivas para su estudio, análisis y comprensión (*Davenport* y *Prusak*, 1998), dichos modelos consideran dos dimensiones del conocimiento: ***La Dimensión Epistemológica*** que destaca la diferencia entre el Conocimiento Explícito (sistemático y fácil de transmitir en forma de datos, fórmulas) y el Conocimiento Tácito (personal y difícil de plantear a través del lenguaje formal, se refiere a visiones subjetivas, intuiciones, juicios) y ***la Dimensión Ontológica***, según la cual la generación del conocimiento debe basarse en el Respaldo organizacional hacia los distintos niveles de entidades creadoras de conocimiento Que pueden ser individual, grupal, por equipos, departamentos y organizacional.

Según Crawford et al. (2006), el proceso de creación del conocimiento se produce por la interacción entre el conocimiento tácito y explícito, generando 4 formas de conversión del conocimiento, como se pude ver en laTabla 1-2:

Tabla 1-2. Proceso de Creación del Conocimiento de *Nonaka-Takeuchi* (Farfán y Garzón, 2006)

ETAPA DEL CICLO	TIPO DE CONVERSION	DESCRIPCION	¿COMO SE LOGRA?	¿QUE RESULTADO GENERA?
Socialización	Tácito a Tácito	Compartir y crear conocimiento tácito a partir de las experiencias.	Caminando, conversando, observando, transfiriendo experiencias.	Conocimiento Armonizado o Compartido

Externalización	Tácito a Explicito	Articular conocimiento tácito a través del dialogo y la reflexión.	Expresar por medio de un lenguaje común. Traducir a conceptos, analogías, metáforas, mapas y modelos.	Conocimiento Conceptual
Combinación	Explícito a Explicito	Sistematización de conceptos con el conocimiento ya almacenado y la información disponible por medio de operaciones mentales colectivas.	Acumular e integrar conocimiento explícito. Transferir y difundir. Editar y publicar conocimiento explícito.	Conocimiento Sistemático
Internalización	Explícito a Tácito	Aprender y adquirir nuevo conocimiento tácito a partir de la práctica.	Aprender conocimiento explicito haciendo o produciendo	Conocimiento Operativo

Formas que producen un proceso de aprendizaje iterativo que dibuja una espiral de transformación de naturaleza dinámica y continua, que ha sido denominada por sus autores *Nonaka-Takeuchi* (1995) como “Ciclo del Conocimiento” (Figura 1-4), donde el tiempo progresa a lo largo de la espiral, y la cantidad de conocimiento aprendido crece con el diámetro de la espiral.

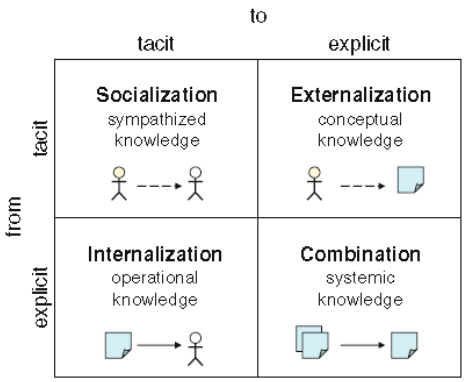


Figura 1-4. Conversión entre el Conocimiento Tácito y Explicito (Schneider, 2009)

En la Tabla 1-3 se presenta un análisis de los modelos de gestión del conocimiento propuestos por diversos autores:

Tabla 1-3. Análisis de los Modelos de Gestión del Conocimiento (Rodríguez, 2006)

	Fundamentación	**Fases**	**Estrategias**
La organización creadora de Conocimiento. (*Nonaka* y *Takeuhci*, 2000)	Transferencia y transformación del conocimiento tácito, y la creación del conocimiento organizacional frente al conocimiento individual	Fases Cíclicas e Infinitas: Compartir conocimiento tácito Crear conceptos Justificar los conceptos Construir un arquetipo Expandir el conocimiento	Creación de mapas de conocimiento, de equipos auto-organizables y de sesiones de dialogo grupal, donde los individuos, mediante esquemas, modelos, metáforas y analogías, revelan y comparten su conocimiento tácito con el resto del grupo.
The 10-Step Road Map.	Diferenciación entre conocimiento tácito	Evaluación de la Infraestructura	Creación de Redes de Comunicación y

(*Tiwana*, 2002)	y explicito, considera también otras clasificaciones del conocimiento, en función de la tipología, focalización, complejidad y caducidad.	Análisis de los Sistema de Gestión del Conocimiento, Diseño y desarrollo Despliegue del Sistema Evaluación de los Resultados	colaboración Trabajo en Equipo
La GC desde una visión Humanística. (De Tena, 2004)	Las personas que conforman la organización, hacen hincapié en la tecnología como la base de un sistema para gestionar el conocimiento.	Consultoría de Dirección Consultoría de Organización Implantación de planes de gestión del conocimiento. Medidas de verificación y seguimiento	Elaboración de Mapas de Conocimiento. Establecimiento de Comunidades de Practica Creación de un almacén de conocimiento Foros de Debate Reuniones Seminarios
La GC desde la Cultura Organizacional. (*Marsal* y *Molina*, 2002)	Fundamentado en el tipo de cultura organizacional existente en la institución	Auto diagnostico Gestión Estratégica Definición y Aplicación del modelo GC. Gestión del cambio. Indicadores para medir el impacto de la GC.	Páginas Amarillas Comunidades de Aprendizaje Buenas Practicas Encuentros de Asistencia y Ayuda
Un Sistema de GC de una	Análisis exhaustivo de la cultura	Definición de un plan de acción para	Círculos de Intercambio de

organización Escolar. (Duran, 2004)	organizacional	generar la cultura adecuada Análisis del Capital Intelectual Análisis de las TIC Creación de un sistema de GC y puesta en marcha de algunas actividades grupales ideadas para la GC	conocimiento. Benchmarking *Knowledge*-Café Técnicas y/o dinámicas grupales.
La Gestión del Conocimiento en Educación. (*Sallis* y *Jones*, 2002)	Cada organización educativa debería poseer y construir su propia estructura, su propio sistema de GC, en función de sus características, sus fortalezas y debilidades.	Clasificación del Conocimiento. Marco de Referencia para la GC. Auditoria del Conocimiento. Medición del Conocimiento. Tecnología y gestión del conocimiento. Explotación del conocimiento.	Mapas de Conocimiento. Creación y Desarrollo de comunidades virtuales Trabajo Colaborativo

A pesar de la existencia de incontables modelos para la gestión del conocimiento, la revisión de algunos de ellos y de la literatura especializada en este ámbito (Davenport y *Prusak*, 2001; *Davenport*, *De Long* y *Brees*, 1997; *Wiig*, 1997; *Rivero*, 2002; *Alavi* y *Leidner*, 1999), nos permite agruparlos en tres tipos según el núcleo, los objetivos, la metodología, los participantes, etc., alrededor del cual se desarrollan (Rodríguez, 2006):

- **Almacenamiento, acceso y transferencia de conocimiento:** modelos que no suelen distinguir el conocimiento de la información y los datos y que lo conciben como una entidad independiente de las personas que lo crean y lo utilizan. Este tipo de modelos de GC se centran en el desarrollo de metodologías, estrategias y técnicas para almacenar el «conocimiento» disponible en la organización en depósitos de fácil acceso para propiciar su posterior transferencia entre los miembros de la organización.

- **Sociocultural:** modelos centrados en el desarrollo de una cultura organizacional adecuada para el desarrollo de procesos de gestión del conocimiento. Intentan promover cambios de actitudes, fomentar confianza, estimular la creatividad, concienciar sobre la importancia y el valor del conocimiento, promover la comunicación y la colaboración entre los miembros de la organización, etc.

- **Tecnológicos:** modelos en los que destaca el desarrollo y la utilización de Sistemas (por ejemplo: *data-warehousing*, intranets, sistemas expertos, sistemas De información, web, etc.) Y herramientas tecnológicas (por ejemplo: motores De búsqueda, herramientas multimedia y de toma de decisiones) para la Gestión del conocimiento.

1.2. METODOLOGIAS TRADICIONALES DE DESARROLLO DE SOFTWARE

El proceso de desarrollo de software no es una tarea fácil, por lo cual existen varias propuestas metodológicas que se involucran en distintas etapas del proceso de desarrollo, entre ellas tenemos las propuestas más tradicionales que se centran especialmente en el control del proceso, e imponen una disciplina de trabajo sobre el proceso de desarrollo de software con el objetivo de asegurar que el producto que se obtenga satisfaga los requerimientos del usuario y reúna estándares aceptables de calidad. Estas propuestas han demostrado ser efectivas y necesarias en un gran número de proyectos, pero también han presentado problemas en muchos otros. (Amaro y

Valverde, 2007). En este capítulo se realiza una corta definición de lo que son las metodologías tradicionales, y se presenta RUP como la más utilizada.

1.2.1. DEFINICION

Las metodologías tradicionales para el desarrollo de software también conocidas como metodologías pesadas, tienen un énfasis en la planificación, en el control del proyecto, en la especificación precisa de requisitos y en el modelado. Mediante una rigurosa definición de roles, actividades, artefactos, herramientas, modelos y documentación detallada abordan el desarrollo de software en proyectos de gran tamaño demostrando ser efectivas y necesarias, sin embargo, las metodologías tradicionales no se adaptan adecuadamente a los cambios, por lo que no son métodos adecuados cuando se trabaja en un entorno, donde los requisitos no pueden predecirse o bien pueden variar (Chau et al., 2003).

1.2.2. RUP

RUP o *Rational Unified Process*, en español Proceso Unificado de *Rational*, fue desarrollado por *Rational Software*, actualmente propiedad de IBM, vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de *Rational* 5.0, es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, y junto con UML constituyen la metodología estándar más utilizada para el análisis, diseño, implementación y documentación de sistemas orientados a objetos. Es un proceso de desarrollo de software, definido como un conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software, es adaptable para proyectos a largo plazo y establece refinamientos sucesivos de una arquitectura ejecutable, se caracteriza por ser (Heredia et al., 2011):

- **Dirigido por casos de uso:** El proceso de desarrollo sigue una trayectoria que avanza a través de los flujos de trabajo generados por los casos de uso. Los casos de uso se especifican y diseñan al principio de cada iteración.

- **Centrado en la arquitectura:** La arquitectura involucra los elementos más significativos del sistema y está influenciada entre otros por las plataformas de software, sistemas operativos, sistemas de gestión de bases de datos, además de otros como sistemas heredados y requerimientos no funcionales.

- **Iterativo e incremental:** El proceso se divide en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y las cuales se definen según el nivel de madurez que alcanzan los productos que se van obteniendo con cada actividad ejecutada.

RUP está dividido en cuatro fases: **Inicio**, define el alcance del proyecto. **Elaboración**, definición, análisis, diseño del producto. **Construcción**, implementación del producto. **Transición**, es el fin del proyecto y puesta en producción del producto. Y define nueve disciplinas a realizar en cada fase del proyecto: **Modelado del negocio**, **Análisis de requisitos, Análisis y diseño, Implementación, Pruebas, Distribución, Gestión de configuración y cambios, Gestión del proyecto y Gestión del entorno** (Heredia et al., 2011). En la Figura 1-5 muestra cómo varía el esfuerzo asociado a las disciplinas según la fase en la que se encuentre el proyecto RUP.

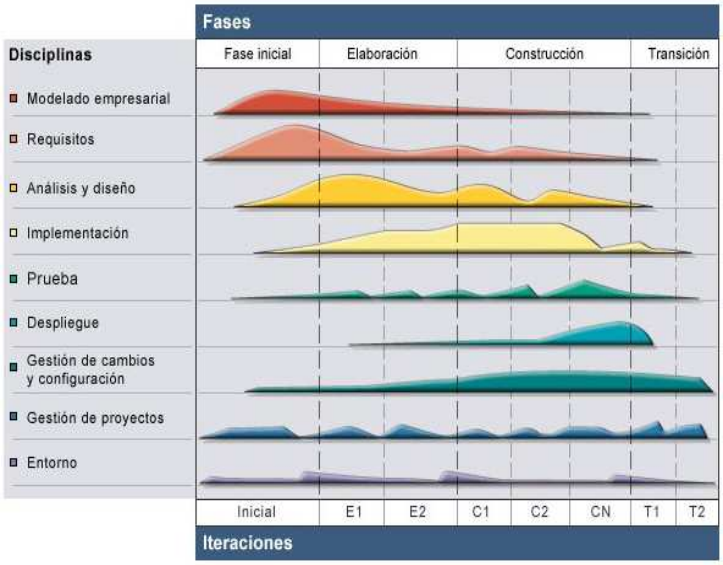


Figura 1-5. Esfuerzo en las Actividades de RUP según la fase del proyecto.

En contraposición, las metodologías ágiles aportan nuevos métodos de trabajo que apuestan por una cantidad apropiada de procesos. Es decir, no se desgastan con una excesiva cantidad de cuestiones administrativas (planificación, control, documentación) ni tampoco defienden la postura extremista de total falta de proceso.

1.3. METODOLOGIAS AGILES DE DESARROLLO DE SOFTWARE

El desarrollo de software ágil ha tenido una gran influencia en cómo se lleva a cabo el desarrollo de software, y se ha convertido en un tema importante para los sistemas de ingeniería y de información de software, presenta una serie de cambios en cómo se planean los desarrollos de software, y como es la comunicación con los clientes con respecto a las metodologías tradicionales (*Dingsøyr* et al., 2010). En este capítulo se ofrece una definición de desarrollo ágil de software a través de un resumen de los estudios existentes, se presenta información del ciclo de vida del desarrollo de software con metodologías agiles, se realiza una comparación con las metodologías tradicionales y

se presentan sus ventajas y desventajas, además de un listado de las metodologías agiles utilizadas en la industria del software.

1.3.1. DEFINICION

Las metodologías ágiles para el desarrollo de software tienen como objetivo esbozar los valores y principios que deberían permitir a los equipos desarrollar software de calidad rápidamente de en forma ágil y eficaz, y responder a los cambios que puedan surgir a lo largo del proyecto (Heredia et al., 2011). Surgen en una reunión celebrada en febrero de 2001 en Utah – USA, donde nace formalmente el término "ágil" aplicado al desarrollo de software., como un solución para todos los problemas que persiguen al desarrollo de proyectos de software desde sus inicios, con el fin de satisfacer los requerimientos cambiantes de los sistemas actuales, pero conservando la calidad del producto (*Bioul et al.*, 2010).

Tras esta reunión se creó *The Agile Alliance*, una organización sin ánimo de lucro dedicada a promover los conceptos relacionados con el desarrollo ágil de software y a ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el **Manifiesto Ágil** (http://agilemanifesto.org/), un documento que resume la filosofía ágil, con la definición de los valores y objetivos del proceso de desarrollo ágil, donde se promueve los individuos y sus interacciones, sobre los procesos y herramientas (*Bioul et al.*, 2010), y cuyas principales ideas son:

Individuos e interacciones sobre procesos y herramientas
Software funcionando sobre documentación extensiva
Colaboración con el cliente sobre negociación contractual
Respuesta ante el cambio sobre seguir un plan

Y se definen los doce principios de las metodologías agiles (Tabla 1-4), que son características que diferencian un proceso ágil de uno tradicional. Los dos primeros principios son generales y resumen gran parte del espíritu ágil, el resto tienen que ver con

el proceso a seguir y con el equipo de desarrollo, en cuanto a metas a seguir y organización del mismo. Los principios son (*Canós et al*., 2003):

Tabla 1-4. Doce Principios de las Metodologías Agiles de Desarrollo de Software (*Canós et al*., 2003)

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

Según Crawford (2005), las metodologías agiles se caracterizan por ser: **Livianas**, puesto que ellas se consideran más fáciles de usar y no enfatizan en la planificación y documentación detallada como sí lo hacen los métodos tradicionales más formales, **Adaptativas**, porque consideran los cambios como una realidad inevitable y no como excepciones, lo que permite una rápida reacción frente al cambio, **Iterativas**, porque dividen el desarrollo del proyecto en ciclos muy cortos. Al final de cada ciclo una porción ejecutable del sistema es entregada al usuario para que éste la valide. Son métodos de la ingeniería de software interactivos e incrementales donde los requerimientos están en constante cambio de acuerdo a las necesidades del cliente. Según Doran (2004), estos métodos siguen el ciclo de vida del desarrollo de software, incluyendo: Elicitación de Requisitos, Análisis, Diseño, Codificación, Pruebas y Entrega del Software, acompañado de un seguimiento y retroalimentación de los clientes (Figura 1-6).

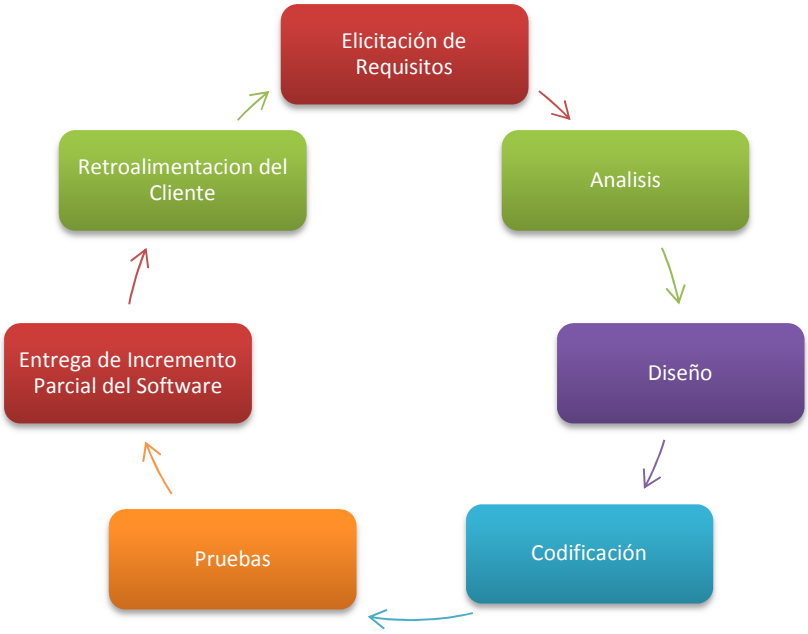


Figura 1-6. Ciclo de Vida del Desarrollo de Software en un Proceso Ágil (Doran, 2004)

Nerur y *Balijepally* (2005) afirman que los métodos ágiles y tradicionales difieren en varios aspectos, incluyendo sus supuestos fundamentales, el enfoque de gestión de los proyectos, la gestión del conocimiento, la asignación de funciones, el papel del cliente, los ciclos del proyecto y el modelo de desarrollo (*Dingsøyr et al.*, 2010). En la Tabla 1-5 *Canós* (2003), recoge las principales diferencias de las metodologías ágiles con respecto a las tradicionales (no ágiles). Estas diferencias afectan no sólo al proceso en sí, sino también al contexto del equipo así como a su organización.

Tabla 1-5. Diferencias entre metodologías agiles y no agiles (*Canós et al.*, 2003).

Metodologías Agiles	Metodologías Tradicionales
Basadas en heurísticas provenientes de prácticas de producción de código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
Especialmente preparados para cambios durante el proyecto	Cierta resistencia a los cambios
Impuestas Internamente (por el Equipo)	Impuestas Externamente
Proceso menos controlado, con pocos principios	Proceso mucho más controlado, con numerosas políticas/normas
No existe contrato tradicional o al menos es bastante flexible	Existe un contrato prefijado
El cliente es parte del equipo de desarrollo	El cliente interactúa con el equipo de desarrollo mediante reuniones
Grupos pequeños (<10 Integrantes) y trabajando en el mismo sitio	Grupos grandes y posiblemente distribuidos
Poco artefactos	Mas artefactos
Pocos roles	Mas roles
Menos énfasis en la arquitectura del software	La arquitectura del software es esencial y se expresa mediante modelos

1.3.2. VENTAJAS Y DESVENTAJAS DE LAS METODOLOGIAS AGILES DE DESARROLLO DE SOFTWARE

Las metodologías ágiles surgen como una necesidad a la hora de satisfacer los cambiantes requerimientos de desarrollo de los sistemas actuales, pero manteniendo la calidad del producto resultante. Como toda metodología, tiene sus ventajas y desventajas, según *Sharman et al.* (2013) las de las metodologías ágiles son:

VENTAJAS

- Adaptación al Entorno Cambiante: En el método de desarrollo ágil de software, el software se desarrolla en varias iteraciones. Cada iteración se caracteriza por el análisis, diseño, implementación y pruebas. Todos los cambios que actualizan el software son bienvenidos por parte del cliente en cualquier etapa de desarrollo.

- Asegura la Satisfacción del Cliente: Esta metodología requiere la participación activa del cliente en todo el desarrollo, así al final lo que se obtiene como el producto final es de alta calidad y asegura la satisfacción del cliente, ya que se desarrolla todo el software basado en los requisitos tomados de cliente.

- Menos Documentación: La documentación de la metodología ágil es breve y al punto, sin embargo, depende del equipo ágil. Las principales cosas que deben estar en la lista de la documentación son las características del producto, la duración de cada iteración y la fecha. Esta breve documentación ahorra tiempo de desarrollo y entregar el proyecto en el menor tiempo posible.

- Reduce los Riesgos del Desarrollo: Como se entrega a los clientes una versión mini del software en cada iteración, después de cada ciclo de desarrollo se recibe la evaluación de los clientes, lo que advierte a los desarrolladores sobre los problemas futuros que pueden ocurrir en las etapas posteriores del desarrollo. También ayuda a descubrir los errores rápidamente y se corrijan de inmediato.

DESVENTAJAS

- La Interacción con el Cliente es el Factor Clave en el Desarrollo de Software de Éxito: Las metodologías agiles se basa en la implicación del cliente ya que todo el proyecto se desarrolla de acuerdo con los requisitos establecidos por los clientes. Así que si el representante de los clientes no es claro acerca de las características del producto, el proceso de desarrollo puede salir mal.

- La Falta de Documentación: A pesar de que la menor documentación ahorra tiempo de desarrollo, por otro lado, es una gran desventaja para el desarrollador, debido a que hay menos información disponible, lo que dificulta que los nuevos desarrolladores que se unan al equipo en una etapa posterior comprendan la metodología seguida para desarrollar el software.

- El Consumo de Tiempo y el Desperdicio de Recursos debido al Cambio Constante de las Necesidades: Si los clientes no están satisfechos con el software parcial desarrollado por cierta iteración y cambian sus necesidades entonces este incremento no sirve de nada, por lo que es un desperdicio total de tiempo, esfuerzo y los recursos necesarios para desarrollar ese incremento.

1.3.3. METODOLOGÍAS AGILES EN LA INDUSTRIA DEL DESARROLLO DE SOFTWARE

Actualmente existen varias metodologías que facilitan el desarrollo de proyectos agiles, algunas se centran en diferentes aspectos del ciclo de vida del desarrollo de software, mientras otras se centran en la gestión del conocimiento (*Sharman et al.*, 2012), y aunque coinciden en los principios enunciados anteriormente, cada metodología tiene características propias y hacen hincapié en algunos aspectos más específicos. A

continuación se resumen algunas metodologías ágiles, la mayoría de ellas ya han sido utilizadas con éxito en proyectos reales (*Canós et al.*, 2003):

1.3.3.1. Feature Driven Development - FDD

FDD o Desarrollo Basado en Funcionalidades, es una metodología ágil para el desarrollo de software iterativa y adaptativa, desarrollada por *Jeff De Luca* y *Peter Coad*, que propone definir una serie de '*Features*' o funcionalidades que debe contener el producto, organizadas en jerarquías, con un alcance lo suficientemente corto como para ser implementadas en un par de semanas (*Bioul*, 2010), se enfoca en iteraciones cortas que entregan funcionalidades visibles para el cliente, y a diferencia de otras metodologías ágiles no cubre todo el ciclo de vida sino sólo las fases de diseño y construcción, se considera adecuado para proyectos mayores y de misión crítica (Lynn, 2011).

1.3.3.2. Crystal Clear Methodologies

Crystal Clear Methodologies, es un conjunto de metodologías para el desarrollo de software, propuesta por *Alistair Cockburn*, está caracterizada por estar centrada en las personas que componen el equipo y la reducción al máximo del número de artefactos producidos, tiene un enfoque ágil, con gran énfasis en la comunicación, y con cierta tolerancia que la hace ideal en los casos en que sea inaplicable la disciplina requerida. *Crystal Clear* maneja iteraciones cortas con *feedback* frecuente por parte de los usuarios/clientes, minimizando de esta forma la necesidad de productos intermedios (*Canós* et al., 2003).Propone distintos procesos a aplicar según tres variables básicas: el tamaño del proyecto, la criticidad y las prioridades del mismo (*Bioul et al.,* 2010)

Los siete valores o propiedades sobre los que se basa *Crystal Clear* son (Amaro y Valverde, 2007):

- **Entrega frecuente:** Consiste en entregar software a los clientes con frecuencia, no solamente en compilar el código.

- **Comunicación osmótica:** Todos juntos en el mismo cuarto.

- **Mejora reflexiva:** Tomarse un pequeño tiempo (unas pocas horas por algunas semanas o una vez al mes) para pensar bien qué se está haciendo, cotejar notas, reflexionar, discutir.

- **Seguridad personal:** Hablar cuando algo molesta, decirle amigablemente al manager que la agenda no es realista, o a un colega que su código necesita mejorarse, o que sería conveniente que se bañase más seguido.

- **Foco:** Saber lo que se está haciendo y tener la tranquilidad y el tiempo para hacerlo. Lo primero debe venir de la comunicación sobre dirección y prioridades, típicamente con el Patrocinador Ejecutivo. Lo segundo, de un ambiente en que la gente no se vea compelida a hacer otras cosas incompatibles.

- **Fácil acceso a usuarios expertos:** Define la importancia del contacto directo con expertos en el desarrollo de un proyecto.

1.3.3.3. eXtreme Programming - XP

XP es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la refactorización o reutilización del código desarrollado,se centra en ofrecer la solución técnica más simple posible que cumpla con los objetivos de negocio (*Lynn*, 2011). Es el método más exitoso de desarrollo de software ágil debido a su enfoque en la satisfacción del cliente, ya que mantienen la máxima interacción posible con este para desarrollar el software. XP divide todo el ciclo de vida de desarrollo de software en varios ciclos de desarrollo cortos, le da la bienvenida e incorpora cambios o

requerimientos de los clientes en cualquier fase del ciclo de vida de desarrollo (*Sharman et al.*, 2012), y se basa en:

- **Pruebas Unitarias:** Se traduce en las pruebas realizadas a los principales procesos, de tal manera que se puede adelantar en algo hacia el futuro, se pueden hacer pruebas de las fallas que pudieran ocurrir obteniendo los posibles errores.

- **Programación en pares:** Una particularidad de esta metodología es que propone la programación en pares, la cual consiste en que dos desarrolladores participen en un proyecto en una misma estación de trabajo. Toda la producción de código debe realizarse con trabajo en parejas de programadores. Esto conlleva ventajas implícitas (menor tasa de errores, mejor diseño, mayor satisfacción de los programadores, etc.).

- **Refactorización:** Es una actividad constante de reestructuración del código con el objetivo de remover duplicación de código, mejorar su legibilidad, simplificarlo y hacerlo más flexible para facilitar los posteriores cambios. Se mejora la estructura interna del código sin alterar su comportamiento externo.

- **Propiedad colectiva del código:** Cualquier programador puede cambiar cualquier parte del código en cualquier momento.

- **Integración continua:** Cada pieza de código es integrada en el sistema una vez que esté lista. Así, el sistema puede llegar a ser integrado y construido varias veces en un mismo día.

1.3.3.4. Dynamic Systems Development Method - DSDM

DSDM es un método de Desarrollo de Sistemas Dinámicos, nace en 1994 con el objetivo de crear una metodología RAD unificada, se caracteriza por ser un proceso iterativo e

incremental donde el equipo de desarrollo y el usuario trabajan juntos (*Canós et al.*, 2003). Se basa en la construcción de una serie de artefactos que permitan obtener una confirmación por parte del usuario de que la solución se ajusta a sus necesidades, antes de iniciar las actividades de desarrollo, está enfocada a proyectos con características RAD (*Rapid Application Development*), con una fase única de estudio de factibilidad y luego una serie de fases iterativas para el análisis, diseño y desarrollo (*Bioul et al.*, 2010).

Guiada por siguientes principios, propone cinco fases: estudio viabilidad, estudio del negocio, modelado funcional, diseño y construcción, y finalmente implementación. Las tres últimas son iterativas, además de existir realimentación a todas las fases (Amaro y Valverde, 2007):

- El involucramiento del usuario es imperativo.
- Los equipos de DSDM deben tener el poder de tomar decisiones.
- El foco está puesto en la entrega frecuente de productos.
- La conformidad con los propósitos del negocio es el criterio esencial para la aceptación de los entregables.
- El desarrollo iterativo e incremental es necesario para converger hacia una correcta solución del negocio.
- Todos los cambios durante el desarrollo son reversibles.
- Los requerimientos están especificados a un alto nivel.
- El *testing* es integrado a través del ciclo de vida.
- Un enfoque colaborativo y cooperativo entre todos los interesados es esencial.

1.3.3.5. Adaptive Software Development - ASD

ASD o Desarrollo Adaptable de Software, fue desarrollada por *Jim Highsmith* y *Sam Bayer* a comienzos de 1990, es una metodología que se basa en la adaptación continua a circunstancias cambiantes, se caracteriza por ser iterativa, orientada a los componentes de software más que a las tareas, y es tolerante a los cambios. Su ciclo de vida tiene tres

fases: **Especulación**, en la cual se da inicio al proyecto, se establecen los principales objetivos y metas del proyecto, y se planifican las características del software. **Colaboración**, es la fase donde se realiza el desarrollo de las características del producto, utilizando desarrollo iterativo con entregas parciales al cliente. **Aprendizaje**, es la fase final del desarrollo, en la cual se revisa la calidad del producto, y se realiza la entrega al cliente, permite identificar lo que se ha aprendido, tanto positivo como negativo (*Canós et al.*, 2003).

1.3.3.6. Lean Software Development

Lean Software Development, desarrollada por Bob Charettes a partir de su experiencia en proyectos con la industria del automóvil en Japón en los años 80 (*Canós et al.*, 2003), es una translación de los principios y prácticas de la industria manufacturera hacia el dominio de la ingeniería de software, se basa en (*Lynn*, 2011):

- Eliminar los desperdicios
- Ampliar el aprendizaje
- Decidir lo más tarde posible
- Reaccionar tan rápido como sea posible
- Potenciar el equipo
- Crear la integridad

Y busca identificar y confirmar el valor de las diversas características del negocio, priorizar aquellas que crean el mayor valor, y hacer el proceso de desarrollo lo más eficiente posible. Incluye también una actividad de retroalimentación, de manera que el equipo reciba críticas regulares de su trabajo, para la mejora continua (*Lynn*, 2011).

1.3.3.7. Kanban

Kanban es una metodología para la gestión de los cambios, utilizada para complementar las metodologías agiles, se basa en *Just-In-Time*, es decir busca que se entregue justo lo necesario en el momento necesario para completar el proceso productivo, se aplica comúnmente en actividades de apoyo y mantenimiento en el que la carga de trabajo es impredecible y las prioridades cambian constantemente. En contraste con *Scrum*, es menos prescriptivo. No incluye iteraciones de desarrollo tales como los *Sprints*, no define los roles, como *Scrum Master*, no define reuniones como imágenes a tamaño natural diarios, retrospectivas o demos. *Kanban* utiliza "Tableros" para presentar información sobre el flujo de trabajo: cualquier persona puede ver el estado del trabajo planificado, en progreso y terminado, la capacidad para asumir trabajo adicional, y una indicación de los impedimentos. Los principales destinatarios de Kanban son: la visualización del flujo de trabajo, la limitación de los trabajos en curso, y la medición del tiempo de espera (Lynn, 2011).

1.3.3.8. Scrum

Scrum es una metodología de gestión de proyectos iterativa e incremental, introducida por *Takeuchi, Degrace, Schwaber*, y otros a finales de 1990. Propone una adaptación continua del plan de proyectos a las circunstancias del mismo, utilizada especialmente en proyectos complejos, donde se necesita obtener resultados en corto tiempo, los requisitos son cambiantes o pocos definidos (Canos et al., 2003). El proyecto se divide en piezas individuales que se ejecutan en un periodo de dos a cuatro semanas llamado “Sprint” en cada una de las cuales se obtiene una nueva versión del producto con nuevas funcionalidades, Cada Sprint se inicia con una reunión de planificaciones, donde los miembros del equipo de desarrollo reciben los requisitos desde el *Product Owner*, y concluye con una Revisión del Sprint, en la que el equipo presenta el trabajo realizado en el sprint, y se realiza una retrospectiva de trabajo realizado para identificar oportunidades de mejora (Lynn, 2011).

En la Figura 1-7 se ilustran los diferentes elementos que componen la metodología *Scrum*:

- **Roles:** *Producto Owner*, quien representa los intereses del cliente, proporciona los requisitos y financia el proyecto. *Team*, es el responsable de desarrollar y probar las funcionalidades del proyecto. *Scrum Master*, es el responsable del proceso de scrum, así como de garantizar que cualquier asunto o problema se resuelva (Schneider, 2009).

- **Documentos:** *Producto Backlog*, listado priorizados de los requisitos del proyecto, los cuales se desarrollan en pequeñas iteraciones llamadas *Sprint*. *Sprint Backlog*, listado de los requisitos más importantes que se pueden trabajar en la iteración actual, es definido por el *Product Owner*, el *Scrum Master* y el Equipo. *Sprint Result*, Listado de requisitos desarrollados durante el *Sprint.* (Schneider, 2009).

- **Reuniones:** *Reunión de Planificación del Sprint*, el equipo, el *product owner* y el *scrum master* definen el *Sprint Backlog*, con base en los requisitos priorizados por el *product owner*, y las estimaciones del equipo. *Reunión Diaria*, durante el sprint, se realiza una reunión de 15 minutos todos los días para el intercambio de información, con el objetivo de determinar el progreso logrado e identificar los problemas que requieren atención, cada miembro del equipo da respuesta a las preguntas: ¿Qué hizo ayer? ¿Qué va a hacer hoy? ¿Qué problemas o inconvenientes se presentaron?

  Al finalizar el Sprint se realiza la Reunión de Revisión del Sprint, el *scrum master*, el equipo y el *product owner* se reúnen para examinar los resultados logrados en la iteración (Schneider, 2009).

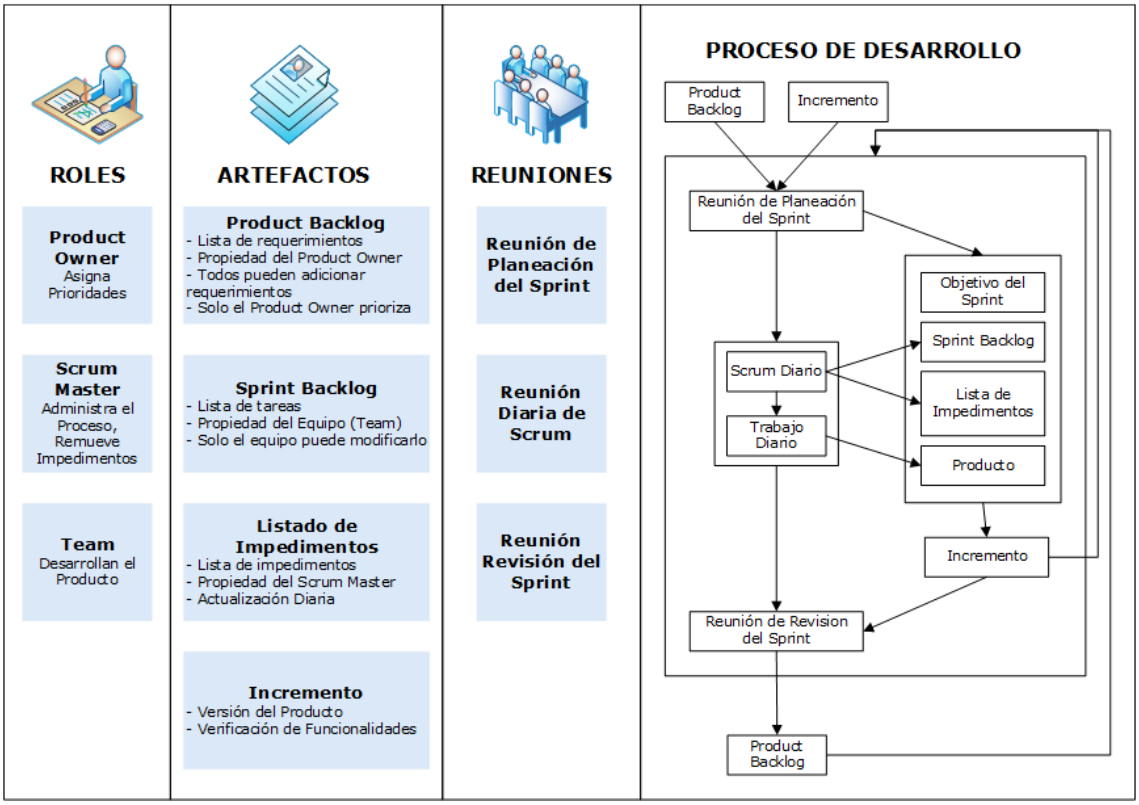


Figura 1-7. Descripción de Roles, artefactos, reuniones y proceso de desarrollo de *Scrum* (Amaro y Valverde, 2007)

1.4. GESTION DEL CONOCIMIENTO EN METODOLOGIAS AGILES DE DESARROLLO DE SOFTWARE

Los programadores poseen un conocimiento valioso sobre el proceso de desarrollo de software, la gestión de proyectos y las tecnologías, generándose así el principal reto de la gestión del Conocimiento en las Metodologías Agiles de Desarrollo de software, específicamente en la transferencia de conocimiento tácito a explícito. En este capítulo se exponen las dificultades de la gestión del conocimiento en proyectos que emplean metodologías ágiles de desarrollo de software, y algunas de las estrategias propuestas por diversos autores para enfrentar dicha problemática.

Históricamente, los enfoques de desarrollo tradicionales usan fuertemente la documentación para capturar el conocimiento de las actividades del ciclo de vida del proyecto, privilegiando la producción de un conjunto amplio de documentos intermedios que le den visibilidad al proceso, mientras que las metodologías agiles enfatizan en el tratamiento del conocimiento tácito sobre el explícito, reemplazando la generación de documentación detallada por una comunicación directa. En la Tabla 1-6, según Crawford et al. (2006), se realiza un comparativo de los artefactos de conocimiento generados por cada una de las metodologías de desarrollo de software Tradicionales o Agiles:

Tabla 1-6. Comparativa Artefactos de Gestión del Conocimiento en las Metodologías de Desarrollo de Software (*Crawford et al.*, 2006)

Estrategias para Compartir el Conocimiento en el Desarrollo de Software		
	Metodologías Tradicionales	**Metodologías Agiles**
Elicitación de Requerimiento y Documentación	Documentos y Artefactos acerca de los requerimientos y el diseño del producto, el proceso de desarrollo, el dominio del negocio y el estado del proyecto.	Documentación Clara y directa tienen significativamente menos documentación fomentan fuertemente la comunicación y la colaboración directa y frecuente, entre los miembros del equipo, y el equipo y el cliente.
Entrenamiento	Se realizan sesiones Formales de Capacitación	Se realizan Prácticas Informales como la programación en parejas y la rotación par
Confianza y Libertad	Interacción entre los miembros de los equipos, es ordenada por un superior, según las necesidades del proyecto	Propiedad colectiva del código, Reuniones stand-up, y programación en parejas. La clave del intercambio de conocimientos aquí son las interacciones entre los miembros

		de los equipos, que es voluntaria y no ordenada por un superior.
Trabajo en Equipo y Roles	Basados en roles, donde cada uno contiene miembros que comparten el mismo papel. Cada rol ejecuta diferentes funciones	Equipos conformados por individuos que realizan todos los roles definidos.

Como se pude ver en la Tabla 1-6, las metodologías agiles reemplazan la mayoría de la documentación que se genera con el enfoque de desarrollo tradicional por la comunicación cara a cara entre los miembros del equipo, basándose en la Socialización para la transferencia del conocimiento, que según *Nonaka* y *Takeuchi* (2000) es el proceso de compartir y crear el conocimiento tácito a través de las experiencias, solo generan y utilizan la información necesaria y significativa que permita mejorar el proceso o el entendimiento del sistema, la cual no es muy detallada y de uso público para todo el equipo (Crawford, 2005), en las metodologías agiles, el intercambio de conocimientos es promovido por varias actividades: la programación en parejas, la rotación par, los clientes localizados en el mismo lugar que el equipo de desarrolladores, las reuniones diarias, los equipos multifuncionales, y las retrospectivas del proyecto, todas ellas basadas en la interacción social, que se postula como el principal medio para que el conocimiento tácito sea compartido (Chau et al., 2003).

Según lo anterior, en la **¡Error! No se encuentra el origen de la referencia.** se realiza un análisis de las diferentes estrategias empleadas en la práctica de metodologías de desarrollo de software ágil, para apoyar el intercambio de conocimiento, y aumentar la conciencia de los participantes acerca de las situaciones que llevan a la extracción del conocimiento tácito, y el uso eficaz de estos conocimientos (*Levy* y *Hazzan*, 2009):

Tabla 1-7. Elemento de Gestión del Conocimiento en Metodologías Agiles (*Chau et al*, 2003 y *Levy* y *Hazzan*, 2009)

Estrategia	Descripción

Documentación	Las metodologías agiles, no eliminan completamente la documentación, solo sugieren que esta se clara, directa y solo la suficiente para facilitar la comunicación y comprensión del sistema. Se recomienda que esta documentación sea pública para todo el equipo.
Requisitos y Conocimientos de Dominio	En lo que respecta a la elicitacion de requisitos y el conocimiento del dominio, las metodologías agiles recomiendan la participación activa de las partes interesadas a través de prácticas que faciliten la colaboración entre los clientes y el equipo de desarrollo en la definición y planificación de las características del sistema a implementar. El conocimiento del sistema y/o dominio se difunde al equipo de desarrollo con mayor eficacia debido al contacto estrecho y frecuente con los clientes.
Aprendizaje Continuo	El aprendizaje continuo con el apoyo de algunos de los métodos ágiles en la forma de retrospectivas, ya que facilitan el aprendizaje de los factores de éxito y los obstáculos de la gestión actual y el proceso de desarrollo.
Espacio de Trabajo Colaborativo	En el desarrollo de software la gestión del conocimiento, se ha vuelto un elemento primordial para el éxito del proyecto, por esto las metodologías agiles proponen utilizar las paredes del espacio de trabajo como medio de comunicación, conformando un espacio informativo y de colaboración, donde el equipo puede encontrar el estado de las tareas personales que pertenecen a la periodo actual del proyecto y las medidas adoptadas. Por lo tanto, todas las partes interesadas del proyecto se pueden actualizar a simple vista en cualquier momento sobre la evolución y el estado del proyecto.
Reuniones Stand-Up	En los procesos de Gestión del Conocimiento, es importante para todo proyecto conocer *"Quien sabe qué"* en el equipo de trabajo o en la organización, para esto las metodologías agiles de desarrollo de software proponen que se realicen reuniones periódicas Stand-Up (Diarias), donde los miembros del equipo presentan el estado de sus tareas de desarrollo desde la última reunión, lo que planea realizar

	durante el día, y de ser necesario también pueden expresar sus consultas durante las reuniones, lo que proporciona visibilidad del trabajo del presentador a sus compañeros y administradores de proyectos, facilitando que el equipo conozca a quién dirigirse cuando tiene que trabajar en las partes del sistema con las que no están familiarizados.
Trabajo Colaborativo con el Cliente.	El trabajo colaborativo con el cliente, es una estrategia definida en los principios establecidos por las metodologías agiles de desarrollo de software, donde se indica:“Los usuarios y desarrolladores deben trabajar juntos diariamente durante el proyecto”, esto con el objetivo de lograr que se comparta la información de una manera más eficiente y eficaz a través de la conversación cara a cara, y la difusión del conocimiento al equipo de desarrollo se facilite dado el estrecho y frecuente contacto con el usuario. El objetivo de esta estrategia es conseguir una retroalimentación constante de los clientes (Crawford, 2005).
Programación en Pares	La programación es pares es una práctica donde dos programadores trabajan frente a una computadora para diseñar, codificar y probar el software en conjunto, lo que facilita la adopción y transferencia de conocimiento, ya que implica que todos los miembros del equipo se familiaricen con el software que se está desarrollando, mejorando así su comprensión del mismo. La utilización de esta práctica trae como ventajas: la disminución de la curva de aprendizaje en un 84%, una mejor comunicación y coordinación entre los miembros del equipo, facilitar el intercambio de conocimientos, y mitiga los riesgos de la pérdida de conocimiento debido a la deserción.

2. DESCRIPCION GENERAL DEL USO DE METODOLOGIAS AGILES EN MVM INGENIERIA DE SOFTWARE S.A.S.

MVM Ingeniería de Software S.A.S. ha basado su fábrica de software en el desarrollo con metodologías tradicionales, pero ha querido incursionar en el uso de las metodologías agiles, como *Scrum*, aplicándolas en proyectos de Desarrollo de Software a la Medida y proyectos de Soporte y Mantenimiento de Software. En este capítulo se hace una descripción del proceso de implementación de las Metodologías Agiles en los diferentes proyectos de MVM Ingeniería de Software.

2.1. METODOLOGIAS AGILES EN PROYECTOS DE DESARROLLO DE SOFTWARE

En MVM Ingeniería de Software para el desarrollo de software, se siguen las mejores prácticas de trabajo colaborativo, que proponen metodologías agiles como es el caso de *Scrum* y de acuerdo con lo propuesto por (*Bioul*, 2010) el objetivo es realizar entregas parciales y definitivas del producto final, priorizadas por el beneficio que aportan al usuario El proceso se inicia con la etapa de **Planeación del Producto** compuesta por los siguientes subprocesos: **Identificación de necesidades del negocio**, en este subproceso el *Product Owner* define el contexto del negocio, los objetivos generales y específicos del producto, y la visión funcional del mismo, basándose en las necesidades insatisfechas del negocio, y la visión del producto definida por el *Sponsor* (Patrocinador). **Definición del proceso ágil**, en el cual se conforma el equipo de trabajo y se asignan sus respectivos roles, se selecciona la metodología ágil a utilizar, los artefactos de trabajo y la técnica de estimación. **Definición del *product backlog***, en este subproceso se identifican las características funcionales del producto, se definen las historias de usuarios, se priorizan y se define el número y tamaño de los *release*.

La **Planeación del Sprint**, se compone de los subprocesos: **Definición de los sprint del *release***, en el cual se definen el *Sprint Backlog*, a partir de las funcionalidades o características contenidas en el *release*, dando prioridad a aquellas funcionalidades que más beneficios aportan al usuario. **Estimación del sprint**, en el cual los miembros del equipo y el *Scrum Máster*, quien es el encargado de velar por que todos los participantes del proyecto sigan los valores y principios de *Scrum*, además de eliminar los obstáculos que impiden que el equipo alcance el objetivo del sprint, utilizando la técnica de estimación "*PlanningPoker*" determinan la duración del sprint, realizan la asignación de los recursos, y establecen los compromisos a cumplir para finalizar el sprint.

Una vez realizada la planeación, se continúa con la etapa de **Ejecución del Sprint** compuesta por los siguientes subprocesos: **Atención de las Historias de Usuario**, en el cual el equipo diseña, construye y prueba cada una de las historias de usuario que conforman el *sprint*. ***Daily meetings***, reunión diaria al final del día, de pie y siempre a la misma hora, entre el *Scrum Máster* y miembros del equipo, donde se evalúa que se hizo en el día anterior, que se hará en el día de hoy, y los impedimentos que se han presentaron.

Finalizado el sprint, se ejecuta la etapa **Cierre del sprint**, donde se llevan a cabo los subprocesos: **Revisión del sprint**, en el cual se hace una entrega del prototipo del producto, generado durante la ejecución del sprint en una reunión con el *Product Owner*, el *Scrum Máster* y todos los miembros del equipos. **Reunión de retrospectiva**, es una reunión en la cual se reúnen todos los miembros del equipo con el *Scrum Máster*, para evaluar la ejecución del sprint, y obtener las lecciones aprendidas acerca de lo que se hizo bien o mal durante el sprint, y cuáles son las acciones por mejorar.

Al finalizar los *sprints del release*, se ejecuta la etapa **Cierre del *release***, compuesto por los siguientes subprocesos: **Aseguramiento de la calidad del producto**, se presenta el producto final a todo el equipo de trabajo, y se realizan pruebas de integración y performance del producto obtenido en el *release*. **Presentación del *release***, en este se

entrega al *Product Owner* y a los usuarios finales, el producto obtenido en la ejecución del *release*, para su retroalimentación.

Descrito el proceso de uso de las metodologías agiles en los proyectos de Desarrollo de Software que implementa MVM Ingeniería de Software, podemos concluir que este se basa completamente en los valores y principios de la metodología *Scrum*, adecuada para este tipo de proyectos. En el próximo capítulo, se realiza una descripción de este proceso en proyectos de Soporte y Mantenimiento de Software.

2.2. METODOLOGIAS AGILES EN PROYECTOS DE SOPORTE Y MANTENIMIENTO DE SOFTWARE

El direccionamiento estratégico de MVM Ingeniería de Software S.A.S. esta soportado en tres pilares: La ingeniería de software, la gestión de proyectos, y la gestión del conocimiento. El modelo de atención y gestión de clientes, se realiza por medio de **Programas de Proyectos**, compuestos por diferentes tipos de trabajo: Proyectos, Bandejas y Requerimientos.

El **Proyecto EPM** es uno de los Programas de Proyectos de MVM Ingeniería de Software S.A.S., compuesto por los siguientes tipos de trabajo: proyectos y bandeja (soporte), los cuales se formalizan por medio de actas de Desarrollo y soporte respectivamente**.** Este proyecto se ejecuta en las unidades del área de tecnología que apoyan el negocio en Empresas Públicas de Medellín, USIN (Unidad de Servicios Informáticos del Negocio) y USII (Unidad de Servicios Informáticos de la Institución), y su alcance esta dado en el soporte de los sistemas de misión crítica de la organización, con aplicaciones que tienen un dinámica de cambio constante, requieren una atención rápida, y un alto conocimiento del negocio, de la organización y la tecnología, y donde alrededor del 80% de los trabajos ejecutados corresponden a tipos de trabajo: bandeja (Soporte).

MVM Ingeniería de Software S.A.S. para la gestión de las bandejas en el programa de proyecto **EPM**, ha implementado metodologías ágiles en el contexto de soporte y

mantenimiento de software, utilizando las buenas prácticas que propone la metodología ***Scrum***, en combinación con la herramienta ***Kanban*** de la metodología Lean, lo que permite adoptar un flujo de trabajo continuo, donde las tareas se pueden realizar por fase, sin iteraciones (Sprint), adecuado para proyectos de mantenimiento con historias de usuario que varían con frecuencia como en este caso.

Con un equipo de trabajo cooperativo y auto-organizado, la gestión del proyecto inicia con la etapa de **Asignación**, donde el cliente indica sus necesidades identificadas por un numero de OC (Orden de Cambio), y son asignadas a los miembros del equipo según sus conocimientos del tema tratado en la necesidad, una vez asignada el miembro del equipo realiza el análisis de la necesidad, la planifica y programa su atención. La etapa de **Ejecución** se realiza a partir de un tablero físico con seis columnas principales, donde se ubican las OC en "*Post-it*" según su estado, como se ilustra en la **¡Error! No se encuentra el origen de la referencia.**, así: ***Inventario Asignadas***, se ubican las OC asignadas a los miembros del equipo que aún no han sido atendidas. ***Análisis y Construcción***, corresponde al listado de las OC que actualmente están en manos del equipo para su desarrollo. ***Pruebas de Codificación***, se ubican aquellas OC que se encuentran en pruebas funcionales por parte de los desarrolladores, para su entrega al usuario. ***Pruebas de Aceptación***, en esta columna se encuentran las OC que están pendientes de validación y aceptación por parte del usuario. ***Producción***, corresponde al listado de las OC que han sido probadas exitosamente por el usuario, pero aún no se ha realizado la aprobación final. ***Cerradas***, es la última columna del flujo de trabajo, contiene el listado de OC que se encuentran aprobadas a satisfacción del cliente. El uso de este tablero le permite al Líder del Proyecto, el monitoreo constante sobre el estado del mismo, y la gestión del balance entre la capacidad y la demanda, es decir le permite identificar si el proyecto cuenta con la capacidad para atender todas las necesidades del cliente, sea que hagan falta recursos o estén sobrando.

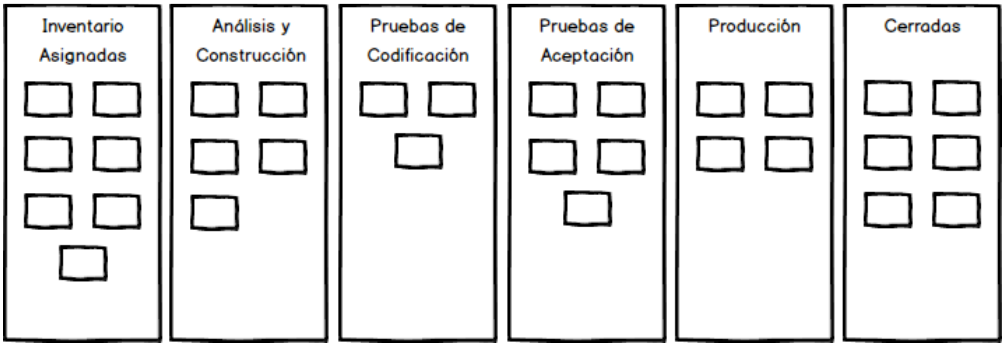


Figura 2-1. Columnas de Gestión del Tablero de Bandejas (Elaboración Propia)

En la Figura 2-2, se ilustran otras dos columnas que hacen parte del Tablero de Gestión del Proyecto, desde las cuales se puede tener un mayor control sobre la ejecución del proyecto, y facilitando la gestión dado que en estas se define una meta de OC a atender en la semana, y se realiza el seguimiento al cumplimiento de esta meta, dichas columnas son: ***Meta Semanal***, en esta columna se hace un listado con las OC que se estima cerrar en la semana, teniendo en cuenta las OC asignadas a los miembros del equipo que aún no se han iniciado, y las OC que se encuentran en desarrollo, y ***Seguimiento y Control***, En esta columna se encuentra la planilla de seguimiento y control de la meta, en esta se indica al final de la semana, la cantidad de OC incluidas en la meta semanal, cuántas de estas se entregaron, cuantas aún están pendientes, y una observación con los motivos del no cumplimiento de la meta.

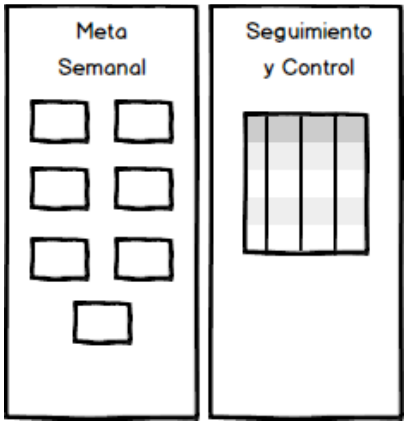


Figura 2-2. Columnas de Control del Tablero de Bandejas (Elaboración Propia)

Durante la etapa de Ejecución, siguiendo las practicas propuestas por *Scrum*, se realizan las **Reuniones Diarias**, en las que los miembros del equipo responden tres preguntas: qué trabajo realizo el día anterior, qué trabajo planea realizar el día de hoy, qué impedimentos se le presentan para realizar su trabajo, con el objetivo de facilitar la transferencia de información y la colaboración, donde cada miembro del equipo inspecciona el trabajo que el resto está realizando para al finalizar la reunión poder hacer las adaptaciones necesarias que permitan cumplir con el compromiso conjunto que el equipo adquirió en la meta semanal.

Como el proceso no es ejecutado por iteraciones como propone *Scrum*, la **Reunión de Retrospectiva** se realiza al final de cada mes, con el objetivo de mejorar la productividad y la calidad del producto entregado, el equipo realiza un análisis de cómo se realizó el trabajo durante el mes, por qué se consiguieron o no los compromisos planteados a partir de las respuesta a las siguientes preguntas: ¿Qué se ha hecho bien? ¿Qué cosas hay que mejorar? ¿Qué se ha aprendido? y ¿Cuáles son los compromisos para el siguiente mes?

3. MODELO DE GESTION DEL CONOCIMIENTO APLICADO EN MVM INGENIERIA DE SOFTWARE

De acuerdo con Gallego (2013), el proceso de Gestión del Conocimiento en MVM tiene como objetivo generar una dinámica de aprendizaje permanente, que permita la adopción, transferencia y el mejoramiento continuo en todos sus activos intelectuales, que se reconocen como sus marcos tecnológicos. La etapa de planeación comprende los subprocesos siguientes: **Identificación de marcos tecnológicos**, donde se definen cuales activos intelectuales deben apropiarse, consolidarse o simplemente explorarse, a partir de la política y estrategia de la organización, en las dimensiones del negocio de los grupos de interés, procesos, herramientas tecnológicas, tendencias y habilidades blandas.

La etapa de hacer, se compone de los subprocesos siguientes: **Recopilar el conocimiento explicito**, a partir de los marcos tecnológicos identificados, logrando así crear base de datos de conocimiento ajustada a las necesidades de la organización, lo que permite rápidamente adoptar conocimiento a partir de los activos de información (documentos, presentaciones, videos, manuales, diagramas, etc.) que tiene la organización. **Diagnóstico de conocimiento**, en el cual se realiza un inventario del conocimiento organizacional, a partir de la valoración en tres dimensiones: autovaloración del colaborador (cada uno valora su nivel de conocimiento por marco tecnológico identificado en el programa de proyectos o área a la cual pertenezca), valoración del líder (cada líder valora el nivel del conocimiento de cada colaborador por cada marco tecnológico identificado en el programa de proyectos o área a la cual pertenezca) y prueba técnica en cada marco tecnológico (cada colaborador debe presentar una prueba en cada uno de los marcos tecnológicos identificados en el programa de proyectos o área a la cual pertenezca). La valoración en estas tres dimensiones permite conocer el nivel de conocimiento de cada colaborador, de acuerdo con el peso definido para cada una de estas, a partir de esta información, se elabora un mapa para identificar el nivel de conocimiento y así saber quién tiene, debe o requiere adoptar, transferir o mejorar su nivel de apropiación en cada uno de los marcos tecnológicos. **Definición de estrategias y**

tácticas para la conversión del conocimiento, es un subproceso que hace parte de la etapa de planeación, el cual está soportado en el ciclo de creación y conversión de conocimiento definido por *Nonaka*, *Toyama* y *Konno* (2000). El objetivo de las estrategias, es cerrar las brechas de conocimiento identificadas en el subproceso de diagnóstico, en tres sentidos: las de adopción de conocimiento, tienen como objetivo la apropiación de los marcos tecnológicos identificados, logrando así, reducir las curvas de aprendizaje, las de transferencia, apuntan a transferencia de conocimiento tácito o explicito que tiene la organización a otros colaboradores, las de mejoramiento continuo, tienen el propósito, la consolidación del conocimiento, que permita la generación de mejoras significativas o desarrollo de nuevos productos o servicios innovadores. Las tácticas son las acciones a llevar a cabo en cada una de las estrategias por medio de actividades de conversión de conocimiento, las cuales se ilustran en la Tabla 3-1.

Tabla 3-1. Actividades y tácticas para la conversión de conocimiento en MVM Ingeniería de Software S.A.S.

Actividad	Objetivo	Táctica
Auto-aprendizaje	Tiene como objetivo la conversión de conocimiento explícito a tácito en cada uno de los marcos tecnológicos de la organización.	Planes de inducción, entrenamiento, semilleros, lecciones aprendidas.
Espacios Conversacionales	Tiene como objetivo la generación de nuevo conocimiento tácito a partir del ya existente en la organización, por medio de los espacios conversacionales	Generación de ideas creativas "R-.creo", Ruta ágil de gestión tecnológica “R-gesto”, lecciones aprendidas, comités técnicos, y planes de entrenamiento
Prototipado	Tiene como objetivo la conversión de conocimiento tácito a explícito. Cuando el	Comités técnicos, elaboración de prototipos, historias de usuario, espacios creativos, partir de las

	conocimiento tácito se convierte en explicito, este se vuelve organizacional y estaría disponible para los demás colaboradores, logrando así una nueva base de conocimientos en cada uno de los marcos tecnológicos definidos	prácticas de pensamiento en diseño, en inglés "*desingthinking*", definidas por Brown (2008).
Documentación	Tiene como objetivo el mejoramiento del conocimiento explícito de marcos tecnológicos existente en la organización.	Elaboración documentos, como diagramas, modelos, manuales, procedimiento, presentación e informes.

En la etapa de verificación se validó la efectividad de las estrategias y tácticas definidas para la Gestión del Conocimiento, lo que permitió la generación, codificación y utilización de este activo intangible, de acuerdo con lo propuesto por *Davenport* y *Prusak* (1997). En la Figura 3-1, se observa todo el proceso de Gestión de Conocimiento, donde se ilustran los componentes principales, de acuerdo a lo anteriormente descrito.

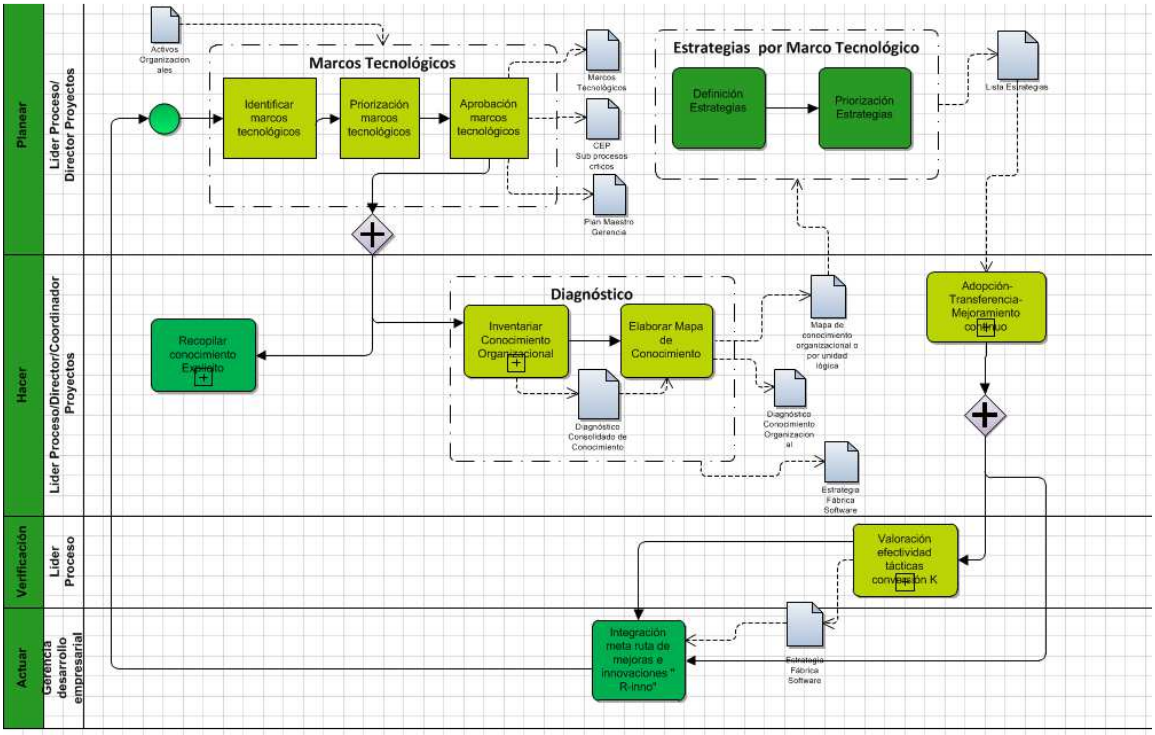


Figura 3-1. Proceso de Gestión de Conocimiento en MVM Ingeniería de Software S.A.S. (Elaboración Propia)

El proceso de Gestión del Conocimiento, está totalmente alineado con los planes de desarrollo de los colaboradores y es así que se asegura el crecimiento y aprendizaje como uno de los objetivos corporativos. Las estrategias y tácticas se registran en los respectivos planes de desarrollo y mejoramiento de cada colaborador, con el objetivo de contribuir con una cultura hacia el conocimiento e innovación, ya que todas las creaciones y nuevos conocimientos se convierten en el motor para el proceso de gestión de la innovación.

4. ESTRATEGIAS PARA LA GESTION DEL CONOCIMIENTO EN PROYECTOS DE DESARROLLO DE SOFWTARE QUE USAN METODOLOGIAS AGILES

A partir de la revisión literaria realizada en el Capítulo 1, la descripción del proceso de implementación de metodologías agiles en el Capítulo 3 y el modelo de gestión del conocimiento en el Capítulo 2, en el presente capítulo se describen las estrategias para asegurar la gestión del conocimiento en proyectos de Soporte y Mantenimiento de Software de ende en MVM Ingeniería de Software S.A.S.

Estas estrategias permitirán fortalecer las actividades de adopción y transferencia de conocimiento por medio de la interacción de los integrantes del equipo en los diversos de escenarios definidos en el proceso de desarrollo de software ágil: asignación, ejecución, reuniones diarias y reuniones de retrospectiva y del uso de los diversos instrumentos definidos en el proceso de gestión del conocimiento: mapas de conocimiento, marco tecnológicos y el Sitio de Gestión de Conocimiento de la organización.

En la Figura 4-1, se ilustran las estrategias para la gestión de conocimiento propuestas, donde se pueden observan sus principales componentes de acuerdo a lo anteriormente descrito. Para cada etapa del proceso, representadas por las columnas en el diagrama, se encuentran las estrategias para la identificación, adopción o transferencia de conocimiento, especificando las actividades a realizar y sus elementos de entradas y salidas. En la etapa de **Asignación**, se propone como estrategia la generación de un ***Inventario de Conocimiento***, que facilitara la identificación del nivel de conocimiento de los miembros del equipo, para definir quién puede transferirlo o debe adoptarlo. En la etapa **Ejecución**, se definen las estrategias ***Mentorias en Campo*** y ***Facilitación Grafica***, con el objetivo de transferir el conocimiento al miembro del equipo que lo requiera. En las **Reuniones Diarias**, se propone la implementación de ***Historias de Conocimiento***, con las cuales se busca emplear tecnologías de información para almacenar el conocimiento

de los expertos, según sea necesario en el proceso. Finalmente en las **Reuniones de Retrospectiva**, se definen dos estrategias que facilitan el mejoramiento continuo del equipo y del proyecto: ***Lecciones Aprendidas*** y ***Mejora Continua***.

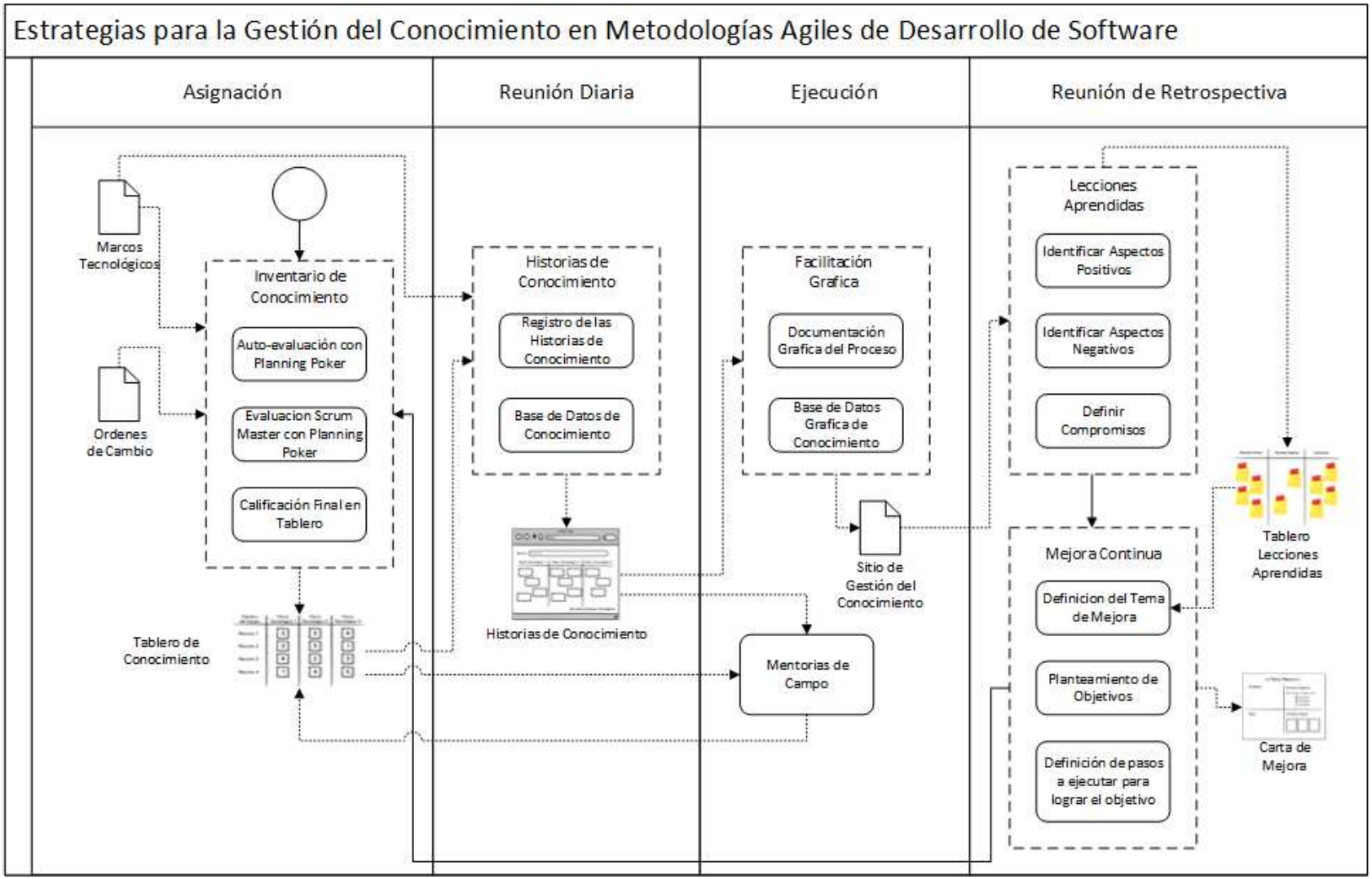


Figura 4-1. Proceso de Gestión del Conocimiento en Proyectos que implementan Metodologías Agiles para el Desarrollo de Software

En la Tabla 4-1, se ilustran las actividades a llevar a cabo en cada una de las estrategias de gestión del conocimientos definidas para el proceso de desarrollo de software a través de metodologías agiles, ilustradas en la Figura 4-1

Tabla 4-1. Estrategias para la Gestión del Conocimiento en el Desarrollo de Software con Metodologías Agiles (Elaboración Propia)

N°	Estrategia	Entradas	Descripción	Medio	Responsable
1	Inventario de Conocimiento	Planeación	En las reuniones de **Planeación** al inicio del sprint, en el caso del programa de proyecto EPM, se debe identificar el nivel de conocimiento de cada uno de los miembros del equipo con respecto a los marcos tecnológicos a trabajar en el proyecto, utilizando como método de auto-evaluación la herramienta de estimación utilizada en el desarrollo de software con metodologías agiles: **PlanningPoker.** Cada uno de los miembros del equipo posee un conjunto de cartas numeradas del 1 al 5, donde 1 es poco o ningún conocimiento y 5 es un conocimiento avanzado. El *Scrum Máster*, comienza enumerando uno a uno los marcos tecnológicos identificados en el proyecto, cada miembro del equipo levanta una tarjeta con la cual indica una calificación como	Tablero de Conocimiento (**¡Error! No se encuentra el origen de la referencia.**)	*Scrum Máster*

			resultado de su auto-evaluación frente al marco tecnológico mencionado, dicha calificación se pone en el tablero, frente al nombre del miembro del equipo, y bajo el marco tecnológico evaluado, finalizada la auto-evaluación, el *Scrum Máster* califica según su percepción cada miembro del equipo en cada marco tecnológico, ubicando su calificación también en el tablero. La calificación definitiva para cada miembro del equipo, se obtiene en un consenso entre el equipo según la auto-evaluación y la calificación del *Scrum Máster*, y se publica en un tablero para conocimiento de todos. La información consignada en el tablero de conocimiento, le permitirá al *Scrum Máster* y a los miembros del equipo, identificar fácilmente quien posee o no el conocimiento, con el fin de definir que debe adoptar el conocimiento, y quien puede transferirlo. Se facilita también la asignación de Historias de Usuario a los miembros del equipo, según sus conocimientos del marco tecnológico.		
2	Historias de	Reuniones	En las reuniones diarias una vez identificados los	Sitio de	Miembro del

Conocimiento	Diarias	marcos tecnológicos a trabajar durante el día, de ser necesario los miembros del equipo que poseen una alta calificación sobre los marcos tecnológicos identificados, debe generar si no existe un registro de **Historias de Conocimiento**, en el Sitio de Gestión del Conocimiento de MVM Ingeniería de Software S.A.S. que permita la transferencia del conocimiento, y facilite la adopción del mismo por parte de los demás miembros del equipo sobre los temas a tratar en la ejecución del soporte, definiendo: • **Marco Tecnológico:** Nombre del Marco Tecnológico, sobre el cual aplica el conocimiento que se va a describir, correspondiente a los Identificados por MVM Ingeniería de Software S.A.S. • **Impacto:** Se define como Alto, Medio o Bajo según la importancia del proceso dentro del negocio.	Gestión del Conocimiento (**¡Error! No se encuentra el origen de la referencia.**)	Equipo

- **Objetivo:** Breve descripción del resultado obtenido al finalizar el proceso, y su beneficios para el proceso.

- **Descripción:** Se realiza una descripción del proceso requerido señalando entradas, salidas, y actividades a ejecutarse para procesar dichas entradas y generar los resultados de salida. Debe ser una descripción completa, clara, y comprensible que facilite la adopción del conocimiento.

La información generada en la **Historia de Conocimiento**, deberá ser registrada en el Sitio de Gestión del Conocimiento de MVM Ingeniería de Software S.A.S., con el fin de que permanezca en el tiempo como Conocimiento Organizacional, y no solo conocimiento del experto, dicho registro se realiza a través de un formulario como el presentado en el **¡Error! No se encuentra el origen de la referencia.**, y se presenta a los usuario para su consulta a través de una pantalla que clasifica las historias de usuario por Marco Tecnológico, y

			presenta principalmente aquellas más populares, más usadas. El sistema también debe permitir al usuario realizar comentarios sobre las historias de usuario sea para complementarlas, actualizarlas o calificarlas. Esta misma estructura de **Historia de Conocimiento** se emplea para dar soluciones alternativas a los impedimentos presentando en la atención de la bandeja con respecto a la tecnología empleada en la atención, se registra la información del impedimento, y las acciones ejecutadas para solucionarlo.		
3	Mentorías	Ejecución	En la atención de las bandejas, si algún miembro del equipo requiere adquirir un nuevo conocimiento, para realizar las actividades de soporte solicitadas por el usuario, un miembro del equipo con mayor conocimiento en el tema brindará un acompañamiento conocido como **Mentoría en Campo**, donde basados en las redes conversacionales, el mentor compartirá su conocimiento al aprendiz, y le brindara apoyo en el	Mentorías en Campo	Miembro del Equipo

			desarrollo de las actividades requeridas para la atención de la necesidad del Cliente, denominada en campo, porque la transferencia del conocimiento se realiza durante la ejecución del proceso.		
4	Facilitación Grafica	Ejecución	La **Facilitación Grafica** es una herramienta que utiliza textos, dibujos simples, palabras, modelos visuales, signos, símbolos y metáforas, para resumir y organizar información importante. Es un método eficaz para interacciones de grupos, y ayudando a reducir la complejidad de una discusión grupal, ya que permite reflejar las diferentes perspectivas de un tema, además de concertar ideas y pensamientos estructuradamente. Por esto los miembros del equipo que poseen el conocimiento, cuando éste se trata de un proceso de negocio, utilizan esta herramienta, para plasmar el conocimiento obtenido durante la ejecución de una actividad de soporte, valiéndose de las imágenes para crear una nueva forma para facilitar el entendimiento del proceso, por aquellas personas que no poseen el conocimiento, ya que el uso de imágenes es una estimulación bilateral del cerebro y	Sitio de KM	Miembro del Equipo

			permite que se adhiera la información prolongadamente en la memoria. Los documentos resultantes de plasmar el conocimiento gráficamente, se almacena en una carpeta organizada por Marco Tecnológico, que posee el conocimiento gráfico de los expertos, dicha carpeta es de conocimiento público del equipo y es almacenada por el Líder, en un lugar de fácil acceso para consulta de todos los miembros del equipo. En el Sitio de Gestión del Conocimiento de MVM Ingeniería de Software S.A.S, se tiene una copia digital del documento generado.		
5	Lecciones Aprendida	Retrospectiva	En las reuniones de Retrospectiva, el líder y los miembros del equipo evalúan lo que ha sido bueno, y lo que podría ser mejor durante el periodo de gestión del proyecto, con el objetivo de mejorar de manera continua su productividad y la calidad del producto que está desarrollando. A partir de esto se propone generar un Tablero de Lecciones Aprendidas, visible para todo el equipo, con el fin de que se tenga presente durante el siguiente periodo	Tablero de Lecciones Aprendidas (**¡Error! No se encuentra el origen de la referencia.**)	Equipo

	de gestión, y se tengan en cuentas las acciones de mejora a ejecutar según los compromisos establecidos. En el tablero se exponen los aspectos **Positivos** y **Negativos** del proyecto durante el periodo, y los **Compromisos** establecidos para el próximo periodo. • **Aspectos Positivos:** En los aspectos positivos, el equipo debe tener en cuenta aquello que sucedió durante el periodo, lo que le pareció bueno, aquello que está funcionando bien, aquello que considere pueda agradecer como una contribución o ayuda, y aquello que debe permanecer durante todos los periodos de gestión del proyecto. • **Aspectos por mejorar:** En los aspectos por mejorar, el equipo debe tener en cuenta aquellas cosas que deben cambiar, con las	

			que no se siente conforme de la gestión del proyecto y aquellas cosas que se pueden mejorar. • **Compromisos:** La columna compromisos son aquellos elementos que se crean a partir de los aspectos positivos y negativos obtenidos de la reunión de retrospectiva, y sobre los cuales se debe trabajar durante el siguiente periodo de gestión del proyecto, ya sea que se continúe trabajando porque se identificó como algo positivo que beneficia el proyecto o porque se debe mejorar o cambiar. En esta columna también pueden incluirse ideas de gestión que pueden probarse, e identificar si generan beneficios para el proyecto. En la siguiente reunión de **Retrospectiva**, el líder del proyecto debe realizar una evaluación del cumplimiento de los compromisos establecidos.		
6	Mejora	Retrospectiva	En la reunión de Retrospectiva, una vez generado el	Mejora	Equipo

Continua	Tablero de Lecciones Aprendidas, a partir de este en una discusión entre los miembros del equipo se selecciona el tema a mejorar durante el periodo, y se dibuja la Carta de Mejora compuesta por: • **Problema:** Cada miembro del equipo tiene dos minutos para consignar en un *post-it*, su punto de vista de la situación actual, y el problema planteado. Finalizados los dos minutos cada uno hace una presentación de sus notas, una vez terminada la puesta en común de las notas, el equipo se da a la tarea de resumir en una frase el problema, la cual es consignada en la Carta de Mejora en el cuadrante “Problema”. • **Meta:** En una discusión con el equipo, sobre como esperan que funcione el proyecto, se define una visión conjunta de la meta a cumplir, en una o dos frases que se consignan en la Carta de Mejora	Continua (**¡Error! No se encuentra el origen de la referencia.**)

- **Nombre del Tema Mejora:** Identificada la situación actual y el problema, y creada la visión de cómo se quiere que sea el proyecto, lo que se quiere modificar o cambiar, se debe definir el nombre del tema, que se ubica en la Carta de Mejora, en la parte superior como título.

- **Próximo Objetivo:** En esta sección se identifica el próximo objetivo a cumplir en camino hacia la meta, se define un intervalo de tiempo, y actividades que componen el cumplimiento del objetivo. Estas deben ser de fácil evaluación, y dejar poco espacio para malas interpretaciones. Se lista en la Carta de Mejora seguidas por un *checkbox*, que permita llevar el control de su cumplimiento.

- **Primeros Pasos:** Se realiza una lluvia de ideas con las acciones que el equipo de trabajo puede ejecutar para lograr el primer objetivo propuesto, estas ideas no tienen

que ser un plan estructurado, son solo unos pequeños pasos para empezar. Una vez definidas las acciones para lograr el objetivo, por votación o discusión, el equipo selecciona tres acciones que considere serán los primeros pasos para lograr la meta, los cuales se ubican en la Carta de Mejora con *post-it*, y son asignadas a los miembros del equipo.

El líder del proyecto es el encargado de realizar un seguimiento a la **Mejora**, verificando el estado de los **Primeros Pasos** dos veces por semana, en mitad de semana (Miércoles) y finalizando la semana (Viernes), después de las reuniones diarias. En la revisión se debe entregar el resultado de la acción para que se marque como ejecutada, si la tarea fue completada se quita de los primeros pasos, y se pone una nueva acción de las identificadas como importantes para lograr el objetivo.

	Al finalizar el periodo de gestión del proyecto (cuando la fecha definida en el objetivo se cumpla), en la reunión de retrospectiva, se evalúa sobre el tema de mejoramiento, identificando si se alcanzó el objetivo, que tanto sea disminuido el problema, que tan cerca estamos de la meta, y si el tema aun es relevante. En caso de que aún sea de interés el tema, se define un próximo objetivo, de lo contrario se genera una nueva Carta de Mejora (*Janlén*, 2013).	

El proceso de Gestión del Conocimiento propuesto, brinda herramientas que faciliten la adopción y transferencia del conocimiento en proyectos que utilizan metodologías agiles para el desarrollo de software, implementando estrategias basadas en redes conversacionales donde prima la comunicación entre los miembros del equipo, pero adicionando una documentación ágil del conocimiento generado a través de herramientas tecnológicos o tableros agiles, aumentando el conocimiento organizacional y garantizando la creación, adopción y transferencia de conocimiento constantemente en el equipo.

CONCLUSIONES

El objetivo principal de un proceso de Gestión del Conocimiento, es generar un ambiente propicio dentro de la organización en el que el conocimiento y la información sean accesibles y puedan ser utilizados para la adquisición del conocimiento, para estimular la innovación y para mejorar la toma de decisiones. En el desarrollo de software la documentación excesiva generada en el proceso de desarrollo con metodologías tradicionales facilitan el proceso de gestión del conocimiento, pero cuando se implementan metodologías agiles donde la mayoría del conocimiento se obtiene y transfiere a través de la experiencia la gestión del conocimiento se hace más compleja, la mayoría de las estrategias para la transferencia del conocimiento y el aprendizaje entre los desarrolladores se basan en la comunicación cara a cara, la colaboración, y la programación en parejas, es decir enfocadas solo en el conocimiento tácito, por esto a través del desarrollo de este proyecto se diseñan estrategias de gestión del conocimiento para el proceso de desarrollo de software ágil que le brinda al proceso herramientas para la adopción y transferencia del conocimiento que permiten la conservación del conocimiento a través del tiempo, y a pesar de la rotación de los miembros del equipo, mediante herramientas tecnológicas o tableros agiles que facilitan cumplir con el objetivo de mantener la información accesible y que pueda ser utilizados para la adquisición del conocimiento, estas estrategias no solo permiten la adopción y transferencia del conocimiento, sino que le permite además llevar a cabo proceso de mejora del que favorecen la adquisición y creacion del conocimiento organizacional.

LECCIONES APRENDIDAS

El desarrollo de este trabajo me permitió conocer y profundizar sobre dos temáticas “Metodologías Agiles” y “Gestión del Conocimiento” que fortalecen mis conocimientos y perfil profesional, brindándome herramientas que me permitieron establecer estrategias que facilitan la gestión del conocimiento en el proceso de desarrollo de software con metodologías agiles, y definir procesos de mejora tanto para el proyecto como para los miembros del equipo.

Por otra parte, el desarrollo de esta investigación me permitió fortalecer mi proceso de formación avanzada por medio de la definición procesos de medición, creación, transferencia y adopción del conocimiento, los cuales se han convertido en factor importante para aumentar la ventaja competitiva de las organizaciones y la importancia

que tienen las redes conversacionales en las metodologías ágiles para la transferencia de conocimiento.

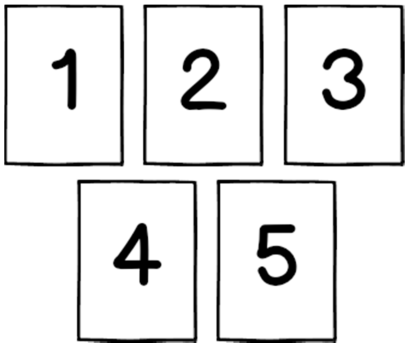
RECOMENDACIONES

La implementación de metodologías agiles en el proceso de desarrollo de software, requiere la ejecución en paralelo de estrategias para la gestión del conocimiento, que permitan que el conocimiento transferido en cada una de las etapa del proceso se convierta un activo organizacional., Es importante también tener en cuenta estrategias que permitan convertir el conocimiento tácito en explicito de una manera ágil por medio de su registro en el Sistema de Gestión del Conocimiento de la organización, con el compromiso de realizar seguimientos y actualización de manera permanente.

ANEXOS

Anexo 1. Tablero de Conocimiento

La estrategia del Tablero del Conocimiento, requiere que se entregue a cada uno de los miembros del equipo un set de 5 cartas, cada una con una número del 1 al 5 que equivalen a la calificación sobre el marco tecnológico evaluado, como se ve a continuación:

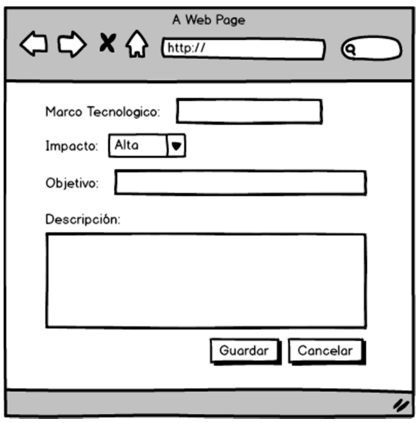


La calificación será consignada en el Tablero de Conocimiento, donde en las filas se registra el nombre del miembro del equipo, y en las columnas se representan los marcos tecnológicos evaluados:

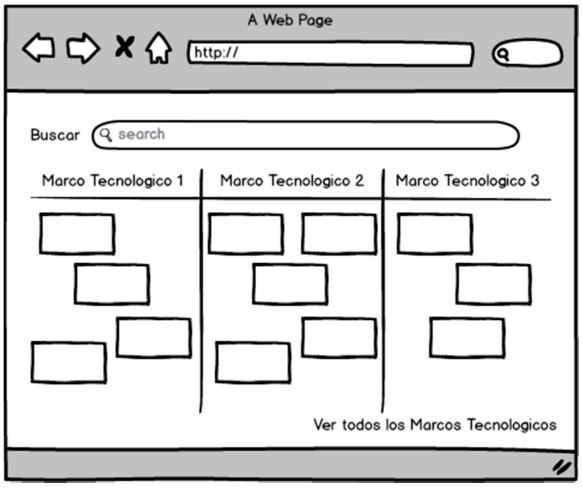
Miembro del Equipo	Marco Tecnologico 1	Marco Tecnologico 2	Marco Tecnologico 3
Recurso 1	3	5	4
Recurso 2	3	5	1
Recurso 3	4	2	3
Recurso 4	1	4	5

Anexo 2. Historias de Usuario

Para el registro de las Historias de Usuario, el Sitio de Gestión del Conocimiento de MVM Ingeniería de Software, brindara a los miembros del equipo un formulario como el que se presenta a continuación, que le permitirá el ingreso de la información definida para la Historia de Usuario:

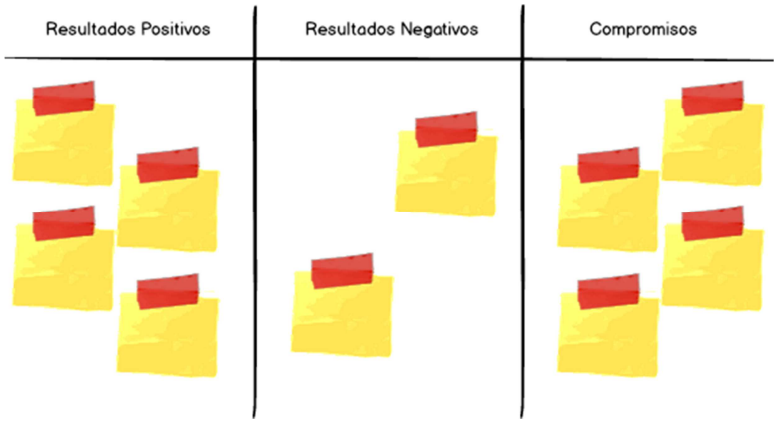


Para la consulta de las Historias de Usuario se define en el Sitio de Gestión del Conocimiento de MVM Ingeniería de Software, el siguiente prototipo, que permitirá a los usuarios visualizar las Historias de Usuario más visitadas organizadas por marco tecnológico, además de realizar consultas sobre todo el sistema de historias de usuario, con el fin de encontrar aquella que se acomode a su necesidad.



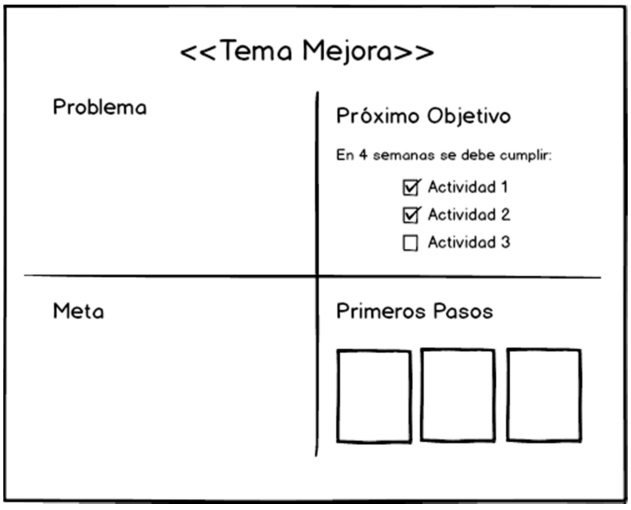
Anexo 3. Lecciones Aprendidas

Prototipo del Tablero a utilizar para realizar la consignación de los elementos obtenidos en la reunión de retrospectiva basados en los aspectos positivos y negativos vistos durante el periodo de gestión del proyecto.



Anexo 4. Carta de Mejora

Prototipo de la Carta de Mejora a generarse en la estrategia de Mejora Continua, para definir el proceso a ejecutarse para la solución del problema identificado en la reunión de retrospectiva.



BIBLIOGRAFIA

Amaro Calderón, S.D. y Valverde Rebaza, J.C. (2007). *"Metodologías Agiles"*. Escuela de Informática. Universidad Nacional de Trujillo. Trujillo – Perú

Bioul, G., Escobar, F., Álvarez, M., Nardin, A., Ricci Aparicio, E. (2010). *"Metodologías Ágiles, análisis de su implementación y nuevas propuestas"*. CACIC 2010 – XVI, Congreso Argentino de Ciencias de la Computación.

Canós, J.H., Letelier, P. y Penadés, M.C. (2003). *"Metodologías Ágiles en el Desarrollo de Software"*.

Chau, T., Maurer, F. y Melnik, G. (2003). *"Knowledge Sharing: Agile Methods vs. Tayloristic Methods"*. Enabling Technologies: Infrastructure for Collaborative Enterprises.WET ICE 2003.Proceedings. Twelfth IEEE International Workshops on, 302 – 307. Recuperado de: http://ieeexplore.ieee.org/

Chen, Mu-Yen y Chen, Chia-Chen (2011). *"Options analysis and knowledge management Implications for theory"*. Information Sciences, 181 (18), 3861–3877. Recuperado de http://www.sciencedirect.com

Chetty, S. (1996). *"The case study method for research in small- and medium - sized firms"*. International Small Business Journal, 5, Octubre – Diciembre.

Crawford Labrin, B. (2004). *"Gestión del Conocimiento en Enfoques de Desarrollo de Software Tradicional y Agilista"*. VI Workshop de Investigadores en Ciencias de la Computación. p. 6-12

Crawford Labrin, B. (2005). *"Métodos Ágiles Enfatizando el Tratamiento del Conocimiento Tácito sobre el Explícito"*. Argentine Symposium on Software Engineering 2005, 127-140.

Crawford Labrin, B., Castro, C. y Monfroy, E. (2006). *"Knowledge Management in Different Software Development Approaches"*. Advances in Information Systems, 4243, 304-313. Recuperado de http://link.springer.com

Davenport, T.H. y Prusak, L. (1997). "*Information Ecology: Mastering the information and knowledge Environment"*. New York: Oxford University Press.

Dermot Doran, H. (2004). *"Agile Knowledge Management in Practice"*. Advances in Learning Software Organizations in Computer Science, 3096, 137-143. Recuperado de http://link.springer.com

Dingsøyr, T., Dybå, T. y Brede Moe, N. (2010). "Agile Software Development: An Introduction and Overview ". Agile Software Development, 1-13.

Farfán Buitrago, D.Y. y Garzón Castrillón, M.A. (2006). "*La gestión del conocimiento".* Universidad del Rosario – Facultad de Administración. Documento de Investigación No. 29, ISSN: 0124-8219.

Gallego, R. (2013). *"La Gestión de Conocimiento e Innovación para la Mejora de Procesos y aprendizaje organizacional: Caso MVM Ingeniería de Software S.A.S"*

Heredia Ruiz, J., Álvarez Almanza, L. y Linares Pons, N. (2011). *"Comparación y tendencias entre metodologías ágiles y formales. Metodología utilizada en el Centro de Informatización para la Gestión de Entidades"*. Serie Científica de la Universidad de las Ciencias Informáticas, 10 (4).
Recuperado por http://publicaciones.uci.cu/index.php/SC.

Holsapple, C.W. y Joshi, K.D. (2002). *"Comprensión de Soluciones de la Gestión del Conocimiento: Evolución de los Modelos de GC en la teoría y en la práctica. Sistemas de Gestión del Conocimiento: teoría y práctica, por Stuart Barnes".* ISBN 84-9732-64-2,269-291.

Janlén, J. (2013). *"Improvement Theme – Simple and Practical Toyota Kata"*. May 14. Recuperado de: http://blog.crisp.se

Kavitha, R.K. y Irfan Ahmed, M.S. (2011).*"A Knowledge Management Framework for Agile Software Development"*. Process Automation, Control and Computing (PACC), 2011 International Conference on, 1–5. Recuperado de: http://ieeexplore.ieee.org/

Lalangui Eras, G.R. (2008). *"Diferentes tipos de conocimiento"*. Enlace: http://galopriva.wordpress.com/2008/07/09/diferentes-tipos-de-conocimiento/

Levy, M. y Hazzan, O. (2009). *"Knowledge Management in Practice: The Case of Agile Software Development"*. Cooperative and Human Aspects on Software Engineering. CHASE '09. ICSE Workshop on, 60-65. Recuperado de: http://ieeexplore.ieee.org/

Lynn Cooke, J. (2011). *"Agile: An Executive Guide*". IT Governance 2011.Recuperado de: http://jim.shamlin.com/study/articles/cooke-agile.html

Nidhra, S., Yanamadala, M., Afzal, W. y Torkar, R. (2013). *"Knowledge transfer challenges and mitigation strategies in global software development—A systematic literature review and industrial validation".* International Journal of Information Management, 33, 333– 355

Nonaka, I. y Takeuchi, H. (2000). *"La organización creadora del conocimiento"*. HBR. Gestión del Conocimiento, 23-49.

Nonaka, I., Toyama, R. y Konno, N. (2000). *"SECI, Ba and Leadership: Unified Model of Dynamic Knowledge Creation"*. Long range planning, 33, 5-34.

Olav Bjørnson, F. y Dingsøyr, T. (2008). *"Knowledge management in software engineering: A systematic review of studied concepts, findings and research methods used"*. Information and Software Technology, 50 (11). 1055-1068.

Pereira Alfaro, H. (2011). *"Implementación de la Gestión del Conocimiento en la Empresa"*. Centro de Gestión de Conocimiento, CEGESTI, 135.

Rodríguez Gómez, D. (2006). *"Modelos para la creación y gestión del conocimiento: Una aproximación teórica"*. Educar, ISSN 0211-819X, 37, 25-39. Recuperado de http://dialnet.unirioja.es

Schneider, K. (2009). *"Experience and Knowledge Management in Software Engineering"*. Springer-Verlag Berlín Heidelberg. Recuperado de http://link.springer.com

Sedera, D. y Gable, G.G. (2010). *"Knowledge Management Competence for Enterprise System Success"*. The Journal of Strategic Information Systems, 19 (4), 296–306. Recuperado de http://www.sciencedirect.com

SegarraCipres, M. y Bou Lusar, J.C. (2004). *"Concepto, Tipos y Dimensiones del Conocimiento: Configuración del Conocimiento Estratégico".*Universitat Jaume I, Castellón. Revista de Economía y Empresa, 2,52 – 53.

Sharma, S., Sarkar, D. y Gupta, D. (2012). *"Agile Processes and Methodologies A Conceptual Study"*. International Journal on Computer Science and Engineering (IJCSE), 4(5).

Wong, K.Y. y Aspinwall, E. (2006).*"Development of a knowledge management initiative and system: A case study"*. Expert Systems with Applications, 30 (4), 633–641. Recuperado de http://www.sciencedirect.com

Yanzer Cabral, A., Blois Ribeiro, M., Lemke, A.P., Tadeu Silva, M., Cristal, M. y Franco, C. (2009). *"A Case Study of Knowledge Management Usage in Agile Software Projects"*. Business Information Processing, 24, 627-638. Recuperado de: http://link.springer.com

Yin, R. K. (1984/1989).*"Case Study Research: Design and Methods, Applied social research"*. Methods Series, Newbury Park CA, Sage