Prácticas de Compartición de Conocimento en los Enfoques Tradicional y Agilista

Broderick Crawford Labrin
Escuela de Ingeniería Informática, Facultad de Ingeniería
Pontificia Universidad Católica de Valparaíso (PUCV)
Av. Brasil 2241, Valparaíso, Chile
Fono 32-273761
broderick.crawford@ucv.cl

Resumen

A la luz de los conceptos básicos de la Gestión del Conocimiento se caracterizan las prácticas esenciales de los enfoques tradicional y agilista de desarrollo de software. Los métodos de desarrollo ágil comparten valores y principios publicados en el Manifiesto Agilista, promoviendo a los individuos y sus interacciones por sobre los procesos y herramientas. Asumiendo las características de cambio acelerado e incertidumbre del actual entorno, enfatizando el tratamiento del conocimiento tácito sobre el explícito reemplazando la generación de documentación detallada por la comunicación cara a cara. Los métodos tradicionales en cambio, usan fuertemente la documentación para capturar conocimiento que se obtiene en cada una de las actividades del ciclo de vida de un proyecto. El crear y compartir conocimiento son relevantes para ambos enfoques de desarrollo de software, por lo que resulta interesante abordarlos desde el punto de vista de la Gestión del Conocimiento, en el presente trabajo se incluyen sus principales conceptos y la relación con los enfoques de desarrollo tradicional y ágil, enfatizando el análisis en una de las actividades más relevantes del ciclo de vida: la elicitación de requerimientos.

Palabras Claves: *Ingeniería de Software, Métodologías Ágiles, Elicitación de Requerimientos, Gestión de la Información y el Conocimiento, Conocimiento Tácito.*

I Workshop de Ingeniería de Software y Bases de Datos

1. Introducción

La Ingeniería de Software es un proceso intensivo de conocimiento en cada una de sus etapas: elicitación de requerimientos, diseño, construcción, testing, implantación y mantención. Hay que considerar también que la implantación de un nuevo sistema de información en una organización puede implicar el rediseño de sus procesos de negocios.

Dado que en ningún proyecto todos los involucrados tienen todo el conocimiento requerido, resulta claro que existirán diversas actividades donde se deberá crear, transferir y difundir conocimiento:

- definición del problema; el cliente y el equipo de desarrollo acuerdan el dominio a considerar ("el negocio")
- elicitación y especificación de requerimientos funcionales y no funcionales del software
- documentación de trabajo del equipo de desarrollo; retención de conocimiento para poder enfrentar cambios en el equipo de desarrollo como también futuros proyectos
- capacitación en el uso del sistema; difusión de conocimiento en la organización
- reunir conocimiento distribuido, creación de bases de datos documentales (procedimientos, mejores prácticas, expertos…) para mejorar procesos internos y capacidad de reacción

Históricamente, los enfoques de desarrollo han facilitado el compartir conocimiento a través de la documentación. La que permite hacer visible el proceso de desarrollo de software a través de la abundante cantidad de documentos que se producen en cada fase. Estos productos intermedios, generalmente documentos de especificación, incluyen modelos del sistema en sus diversos estados de avance. Estados de avance que son planificados en detalle para cada fase en función de los resultados de las fases previas a lo largo del ciclo de vida, privilegiándose un enfoque de proceso basado en tareas y guiado por un plan.

Actualmente, se puede observar un amplio espectro de diferentes formas de abordar un proceso de desarrollo de software, en cuyos extremos destacan por un lado los conservadores y formalistas, quienes se apegan a un proceso de desarrollo tradicional bien planificado, haciendo énfasis en los detalles para asegurar la calidad de sus productos, aspirando a niveles de control y repetición. Y en el otro extremo, el manifiesto agilista [1] que valora más las respuestas a los cambios, a través del uso de fuertes interacciones con los usuarios y entre los desarrolladores, que el seguimiento estricto a una planificación. Argumentan que la rigidez de los planes o contratos muchas veces deriva en que cuando el sistema es entregado el problema que se pretendía resolver cambió o ya no existe.

La existencia de estas dos visiones tiene mucho que ver con la transición de la economía industrial al nuevo paradigma tecno-económico actual, sociedad del conocimiento o como se le denomine al actual entorno competitivo. Donde la manera de producir y entregar bienes y servicios está cambiando [2]. Cuando se compara métodos ágiles y tradicionales, hay autores que sugieren utilizar el término *enfoques Taylorianos* para referirse a las metodologías tradicionales [3, 4]. Argumentando que otros términos también utilizados para referenciarlos, como *métodos planificados o guiados por un plan* ("plan-driven"), *basados en tareas* ("task-based"), o *disciplinados*, no son los más adecuados. Ya que los métodos ágiles también son disciplinados e involucran tareas de planificación, la diferencia radica en el alcance de esta planificación, no en su existencia o no. En el caso de los enfoques Taylorianos la planificación es para todo el proyecto de desarrollo y en el caso agilista es por iteraciones o ciclos más cortos.

Los métodos Taylorianos usan fuertemente la documentación para capturar conocimiento que se obtiene en cada una de las actividades del ciclo de vida del proyecto. Generándose una serie de productos intermedios que intentan asegurar la conformidad con lo planificado. Los métodos ágiles

en cambio, intentan trabajar con el mínimo de documentación necesaria, reemplazándola por la comunicación cara a cara entre los desarrolladores y los usuarios, privilegiando la socialización y el conocimiento tácito sobre el explícito.

El crear y compartir conocimiento son relevantes para ambos enfoques de desarrollo de software, por lo que es interesante abordarlos desde el punto de vista de la Gestión del Conocimiento (GC). En el presente trabajo se incluyen los principales conceptos de la GC y su relación con los enfoques de desarrollo Tayloriano y Agilista enfatizando el análisis en una de las actividades más relevantes del ciclo de vida: la elicitación y especificación de requerimientos.

2. Marco General de Gestión del Conocimiento

En la actualidad nos encontramos en una era denominada la "Sociedad del Conocimiento" caracterizada principalmente por la aparición continua de nuevo saber, con amplias posibilidades de divulgación y donde la empresa pasa a ser concebida desde la idea de un "conjunto de activos tangibles organizados en un determinado proceso productivo para lograr unos objetivos concretos", hasta su consideración actual como "conjunto de activos intangibles, generadores de un capital intangible o capital intelectual"[5].

Con la finalidad de poder aprovechar y obtener mayores ventajas del conocimiento organizacional es que nace la GC, en adelante se entenderá por gestión "el proceso por el cual se obtiene, despliega o utiliza una variedad de recursos básicos (en este caso recursos intangibles como el conocimiento) para apoyar los objetivos de la organización" [6]. La GC es un proceso a través del cual las organizaciones logran descubrir, utilizar y mantener el conocimiento que poseen, con la idea de alinearlo con las estrategias de negocio para la obtención de ventajas competitivas. Debido a que debe contribuir a lograr los resultados comerciales es considerada una disciplina transversal que involucra procesos, personas y tecnologías en una integración constante que implica nuevos métodos de trabajo, preparación a los trabajadores para el cambio y tecnologías que apoyan estas actividades. Para llevar a cabo la gestión del conocimiento se necesita, entre otras cosas, identificar los individuos que poseen el conocimiento y las competencias que les permiten utilizarlo de la mejor manera.

En este artículo se consideran los modelos de Nonaka y Takeuchi, y el modelo Intelect, los cuales plantean que las personas juegan un rol importante y determinante en la generación de valor de las organizaciones, debiendo estas últimas crear las estructuras necesarias para la generación de ventajas competitivas. Es así, que la gestión de los recursos intangibles tiene como objetivo convertir el capital humano (habilidades, conocimientos, etc.) en capital estructural (conocimiento organizacional o lo que queda cuando los empleados se van a sus casas), reduciendo el riesgo de perder valioso conocimiento si la gente deja la organización. De esta, manera lo importante no es retener a las personas, sino que mantener y mejorar los sistemas de manera de poder continuar generando conocimiento.

2.1 Modelo de Creación de Conocimiento (Nonaka y Takeuchi, 1995)

El modelo considera dos dimensiones: 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. La creación del conocimiento se produce por un proceso social de interacción entre el conocimiento tácito y explícito, generando 4 formas de conversión del conocimiento: Socialización (tácito a tácito), Externalización (tácito a explícito), Combinación (explícito a explícito) e Interiorización (explícito a tácito) las que producen un espiral de transformación de naturaleza dinámica y continua, la que ha sido denominada por sus autores como “Ciclo del Conocimiento” [7].

2.2 Modelo de Medición del Capital Intelectual: Intelect (Euroforum, 1998)

El objetivo del modelo es llegar a una exposición clara y práctica de los factores que condicionan la capacidad del aprendizaje en la organización y los resultados esperados de éste. Considerando que el aprendizaje organizacional se basa en un grupo de individuos que expanden continuamente sus aptitudes para crear los resultados que desean cultivando nuevos y expansivos patrones de pensamiento, se proponen elementos como la cultura organizacional, las actitudes, capacidad de trabajo en equipo y otros, como factores básicos para el aprendizaje organizacional. Para cumplir su objetivo estructura tres bloques que representan factores que condicionan la capacidad de aprender, ellos son: (1) Compromiso firme y conciente de toda la organización (en especial de sus líderes), (2) Comportamientos y mecanismos de aprendizaje a todos los niveles (mecanismos de creación, captación, almacenamiento, transmisión e interpretación del conocimiento) y (3) Desarrollo de condiciones organizativas favorables para el aprendizaje y el cambio (que no existan obstáculos para el aprendizaje organizacional, el desarrollo personal, la comunicación al interior y con el entorno). Finalmente, para medir la efectividad de los resultados, se plantea que la capacidad para aprender de la organización se debería concretar en: (1) la posibilidad de evolucionar permanentemente (flexibilidad), (2) una mejora en la calidad de sus resultados, y (3) la empresa se hace más conciente de su integración en sistemas más amplios y produce una mayor relación con su entorno y desarrollo [8].

2.3 Enfoques de Gestión del Conocimiento

Estos modelos demuestran la importancia que tienen los individuos en la generación de valor, proponiendo obtener mejoras en las labores individuales a través del aprendizaje y creación de nuevo conocimiento, las cuales al ser incorporadas a la estructura organizativa (por ejemplo creación de nuevos sistemas de información y procesos), generarán elementos diferenciadores, es decir, ventajas competitivas. Como la Ingeniería de Software es un proceso intensivo de conocimiento, que abarca la captura de requerimientos, diseño, desarrollo, testing, implantación y mantención. Generalmente a partir de un complejo esquema de comunicación en el que interactúan usuarios y desarrolladores, el usuario brinda una concepción de la funcionalidad esperada y el desarrollador especifica esta funcionalidad a partir de esta primera concepción mediante aproximaciones sucesivas. Este ambiente de interacción motiva la búsqueda de estrategias robustas para garantizar que los requisitos del usuario serán descubiertos con precisión y que además serán expresados en una forma correcta y sin ambigüedad, y que sea verificable, trazable y modificable. El crear y compartir conocimiento son relevantes para cualquier enfoque de desarrollo de software, los enfoques de desarrollo tradicionales han facilitado el compartir conocimiento a través de documentación intensiva, pero actualmente se puede observar numerosos métodos que valoran más las respuestas a los cambios, a través del uso de fuertes interacciones con usuarios, que el seguimiento estricto a una planificación: los métodos ágiles.

Los métodos, herramientas y las actuales implementaciones de Gestión del Conocimiento en muchas compañías han seguido también estos dos caminos: documentación intensiva e interacciones. En GC estos enfoques se han denominado Enfoque Centrado en el Producto y Enfoque Centrado en el Proceso.

El Enfoque Centrado en el Producto, tal como el enfoque de desarrollo Tayloriano, se basa en la documentación del conocimiento: creación de documentos, repositorios y posibilidad de reutilización. También se conoce como enfoque centrado en el contenido o "codificación". Considera que el conocimiento es una "cosa" que puede ser manipulada como un objeto independiente. Que es posible capturar, distribuir, medir y gestionar.

El Enfoque Centrado en el Proceso, al igual que los Métodos Agiles, principalmente considera la GC como un proceso de comunicación y colaboración. Donde el conocimiento está estrechamente relacionado con la persona que lo desarrolla y es compartido con otros a través de contactos personales. También se le conoce como enfoque de colaboración o "personalización". 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 [9].

La elección del enfoque más apropiado, o los intentos de combinarlos, dependerán de las características del proyecto, las personas y la organización [10].

3. Métodos Ágiles

Los métodos ágiles son también denominados livianos (lightweight), adaptativos e iterativos.

- Livianos puesto que ellos se consideran más fáciles de usar y no enfatizan la planificación y documentación detallada como sí lo hacen los métodos tradicionales más formales, que en contraste con las ágiles se denominan pesados (heavyweight).
- Adaptativos porque consideran los cambios como una realidad inevitable y no como excepciones. Los métodos ágiles permiten una rápida reacción frente a estos.
- Iterativos 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.

Las metodologías ágiles más representativas son [11]:

- Programación Extrema XP (Extreme Programming),
- Open Source
- Crystal de Cockburn
- Desarrollo de Software Adaptable de Highsmith
- Scrum
- Desarrollo Guiado por Aspectos
- DSDM (Método de Desarrollo de Sistema Dinámico)

Para evitar confusión sobre el significado de "proceso ágil", diecisiete investigadores de metodologías de desarrollo acordaron en el año 2001 a qué denominar agilidad, el resultado de este acuerdo fue la formación de la Alianza Agilista y la publicación de su manifiesto [1].

3.1 Principios del Manifiesto Agilista

El manifiesto de la Alianza Agilista es una definición resumida de los valores y objetivos del proceso de desarrollo ágil, este manifiesto detalla los principios comunes para todos los procesos denominados ágiles.

Los principios son los siguientes:

- Nuestra prioridad más alta es satisfacer al cliente con la entrega temprana y continua de software que pueda valorar.
- Los cambios en los requerimientos son bien aceptados, aún en fases tardías.
- Entregue software trabajando con frecuencia
- Usuarios y desarrolladores deben trabajar juntos diariamente durante el proyecto.
- Construya los proyectos alrededor de individuos motivados. Déles el ambiente y apóyelos en lo que necesiten. Y confíe en ellos.
- El método más eficiente y eficaz de compartir la información es la conversación cara a cara.
- El software trabajando (ejecutable) es la principal medida de avance.
- Los procesos ágiles promueven el desarrollo sostenible.
- Preocupación permanente por la excelencia técnica y el buen diseño refuerzan la agilidad.
- La simplicidad es esencial.
- Las mejores arquitecturas, requerimientos y diseños surgen de los equipos dentro de la organización.
- Los equipos de proyecto evalúan su efectividad a intervalos regulares y ajustan su comportamiento de acuerdo a esto.

3.2 Las metodologías Ágiles versus las Tradicionales

Las metodologías tradicionales imponen una disciplina de trabajo sobre el proceso de desarrollo de software, con el objetivo de asegurar que el software que se obtenga satisfaga los requerimientos del usuario y reúna estándares aceptables de calidad. El trabajo de planificación es riguroso, aún cuando en la práctica muchas veces estas planificaciones no se respetan. 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. Ya que se tiene conciencia de que se producirán cambios, lo que se pretende es reducir el costo de rehacer el trabajo.

Se identifican como principales diferencias entre ambos enfoques las siguientes [12]:

- Las metodologías ágiles son adaptativas más que predictivas. Una metodología tradicional potencia la planificación detallada y de largo alcance de prácticamente todo el desarrollo de software (ejemplo Modelo Cascada). En contraste, las metodologías ágiles proponen procesos que se adaptan y progresan con el cambio, llegando incluso hasta el punto de cambiar ellos mismos.
- Las metodologías ágiles están orientadas más a los desarrolladores que a los procesos de desarrollo. Intentan entonces trabajar con la naturaleza de las personas (desarrolladores y usuarios) asignadas a un proyecto. Permitiendo que las actividades de desarrollo de software se conviertan en una actividad de colaboración grata e interesante.

Se presentan algunas limitaciones a la aplicación de los procesos ágiles en algunos tipos de proyectos. Los agilistas y sus críticos focalizan la discusión en torno a temas como la documentación y codificación. Por un lado se sostiene que el código es el único entregable que realmente importa, desplazando el rol del análisis y diseño en la creación de software. Los críticos precisan que el énfasis en el código puede conducir a la pérdida de la memoria corporativa o conocimiento organizacional, porque hay poca documentación y modelos para apoyar la creación y evolución de sistemas complejos [13]. Independiente de la posición que se adopte, es válido preguntarse respecto a cuáles son las prácticas más adecuadas en el desarrollo de software que permiten aprovechar y obtener mayores ventajas del Conocimiento Organizacional.

4. Gestión del Conocimiento en Métodos Ágiles y Tradicionales

Hace años escuchamos respecto a la crisis del software y nuevamente nos encontramos con sus mismos síntomas: proyectos que no cumplen sus presupuestos y exceden sus plazos de entrega. En muchos casos la causa es la falta de acuciosidad en determinar los requerimientos y controlar sus cambios a lo largo del proceso de desarrollo. Por esto, interesa conocer las prácticas de compartición de conocimiento que se produce durante la elicitación y especificación de requerimientos utilizadas en los enfoques Taylorianos y Agiles.

4.1 Relevancia de la Elicitación de Requerimientos

La parte más difícil de construir un sistema es decidir qué construir. Ninguna otra actividad de este proceso creativo es tan vital como el establecimiento detallado de sus requerimientos, tan significativa en sus resultados finales y tan difícil de corregir. Donde el desarrollador en un proceso de extracción y refinamiento sucesivo intenta capturar los requerimientos a partir de la interacción con el usuario. La inestabilidad de los requerimientos se puede atribuir a que los usuarios no saben lo que quieren, la explicación puede surgir atendiendo un tipo de usuario business man, cuyo mindset es: “give me what I say I want, then I can tell you what I really want” [14]. Muchos usuarios no reconocen sus reales necesidades sin antes ver un componente del sistema, lo que agregado a un entorno de cambio acelerado, hacen de los métodos ágiles y su estrategia de liberar frecuentemente pequeñas porciones de código trabajando una buena alternativa para el mejor entendimiento de las necesidades de los usuarios y su grado de satisfacción por parte del sistema. Lo que parece confirmar que las necesidades del usuario y su solución muchas veces son tácitas y difíciles de explicitar.

En muchos proyectos tradicionales se observa que una completa especificación de requerimientos declarada inicialmente, costosa en tiempo y “papel”, una vez finalizada la construcción puede no satisfacer las reales necesidades. Lo que puede deberse a la falta de contacto entre los usuarios y el equipo desarrollador en las fases de diseño y construcción, como también a que la problemática del negocio haya cambiado desde que se congelaron los requerimientos al final del análisis.

Aunque en la mayoría de los casos los problemas a ser resueltos son mejor entendidos por los usuarios, dado su conocimiento del negocio, la complejidad adicional que surge cuando el usuario no sabe qué quiere, dificulta la comunicación entre usuarios y desarrolladores debiéndose considerar representaciones o modelos que se entiendan fácilmente, de lo contrario la generación de conocimiento será ineficiente. De entre las diferentes formas de representar el sistema hay que elegir la que sea capaz de representar la mayor cantidad de conocimiento. Como en muchos casos es imposible especificar completamente, precisamente y correctamente los requerimientos del producto de software antes de desarrollar alguna versión, desde hace ya muchos años la esencia de este problema se ha tratado de enfrentar haciendo uso de prototipos. Considerando que la mayoría de los usuarios se relacionan mejor con un prototipo de pantalla que leyendo un documento de

requerimientos. Los agilistas plantean que el software funcionando genera mejor y más rápido conocimiento, por lo cual privilegian el uso de iteraciones muy cortas y la refactorización. En cambio, el modelo de desarrollo tradicional (secuencial o cascada), en general no proporciona este feedback. Por lo que también se le considera un modelo determinístico, ya supone que los detalles del proyecto pueden ser completamente determinados desde su inicio.

Existe una percepción de mayor control y predictibilidad, pero debido a las actuales características del entorno y convergencia hacia Internet, la forma tradicional de abordar un proyecto de desarrollo de software ha cambiado, principalmente porque hoy se debe considerar la evolución de los requerimientos a través de todo el ciclo de vida. Los procesos ágiles de desarrollo de software tratan de resolver este problema, es decir, desarrollar software en "los tiempos de Internet" caracterizado por una velocidad de cambio nunca antes vista. Los enfoques ágiles utilizan procesos técnicos y de gestión cuyo objetivo es adaptarse continuamente a los cambios que surgen del conocimiento generado en conjunto con el usuario durante el proceso de desarrollo. Los seguidores de los métodos más formales no pueden entender cómo se podría construir algo que trabaje bien sin analizar los requerimientos. Pero los agilistas no se conforman con perder tiempo valioso y escaso en analizar requerimientos que probablemente cambiarán [15].

Hay un aspecto común relevante en los agilistas: las iteraciones. Una iteración es un incremento de software que es diseñado, programado, testeado, integrado y liberado durante un corto período de tiempo. Una iteración produce una porción ejecutable del producto final, producto que mejorará con futuras iteraciones. Esta forma de trabajar aumenta considerablemente el feedback, comunicación y colaboración, entre usuarios y desarrolladores. El testing es incluído tempranamente, los ambientes de hardware y software son probados desde el comienzo. Y todos los problemas en general son descubiertos a tiempo con este enfoque tolerante a los cambios de requerimientos [16].

4.2. Tratamiento de la Documentación

Todo proceso de desarrollo de software debe ser capaz de capturar y divulgar conocimiento relativo a los requerimientos del producto a desarrollar, dominio del problema y los procesos de negocio que soporta. Este conocimiento los enfoques Taylorianos lo externalizan en un gran número de documentos cuyo objetivo principal es darle visibilidad al proceso de desarrollo y asegurar que los requerimientos, el diseño, la construcción, la mantención y también la gestión del proyecto sean bien entendidos y ejecutados por los involucrados. Aunque depende de la metodología utilizada, esta documentación en general consiste de productos intermedios y entregables al usuario en cada una de las etapas principales consideradas en el proceso de desarrollo. También existen productos no entregables que son propios de los equipos de desarrollo.

Una ventaja de la externalización de conocimiento a través de la documentación es que se reducen las posibilidades de pérdida de conocimiento como resultado del personal que abandona el proyecto y se mejoran las posibilidades de atender proyectos futuros. También permite la colaboración de los equipos de trabajo distribuidos temporal o físicamente.

Por otro lado, hay una gran “cantidad” de conocimiento que es tácito, y que es difícil hacerlo explícito, y muchas veces poco de este conocimiento explícito puede ser documentado en detalle debido a que los equipos de desarrollo perciben esta tarea como larga y tediosa, sin lograr apreciar sus beneficios. Aún, si fuera posible documentar este conocimiento, queda pendiente la manera de asegurar su actualización. Así, los métodos ágiles promueven sólo la generación y uso de documentación necesaria y significativa, que mejore la comunicación o entendimiento del sistema. Con modelos no muy detallados y de uso público para todo el equipo de desarrollo, facilitando la

difusión del conocimiento. El uso de estándares de codificación y modelamiento que promueven las metodologías ágiles también facilitan la transferencia de conocimiento al evitar las reuniones de definiciones, acuerdos o interpretaciones respecto a las herramientas utilizadas. XP, Scrum y otros también consideran la propiedad colectiva de los modelos, lo que facilita su permanente y "ágil" actualización.

Sin embargo, y consistentes con su filosofía de mantener documentación significativa y valiosa, los métodos ágiles declaran preocuparse de mantener actualizados los modelos sólo si el costo de utilizar un modelo desactualizado es mayor al de mantenerlo actualizado. Pero como estos costos son difíciles de tratar, y el uso de un modelo desactualizado no es consistente con una práctica adecuada de compartición de conocimiento, en general los modelos tienden en la práctica a mantenerse actualizados.

Lo anterior se ve reforzado también por el hecho indiscutido que los métodos ágiles son significativamente más livianos en términos de documentación que los Taylorianos, lo que implica esfuerzos de actualización menores. Para compensar la reducción de conocimiento explícito y documentación, los métodos ágiles practican la permanente socialización (comunicación y colaboración) de los desarrolladores y usuarios apuntando a enriquecer el conocimiento tácito. Lo que funciona en equipos pequeños y generalmente colocalizados. En equipos dispersos geográficamente o de muchas personas, la comunicación cara a cara no funciona, y la documentación se hace necesaria para poder compartir conocimiento.

4.3. Ventajas de la Co-localización del Equipo

En relación a la elicitación de requerimientos, los métodos ágiles propugnan la activa participación de los usuarios en el mismo lugar de trabajo y por períodos largos de tiempo. Respecto a las prácticas utilizadas, éstas no difieren significativamente de las utilizadas en algunos proyectos tradicionales (Entrevistas, Casos de Uso, Storyboard, Brainstorming, Joint Application Design (JAD), Focus Groups [17]. La diferencia más significativa radica en que los métodos Taylorianos no promueven particularmente las prácticas que aseguren la participación del usuario. En este aspecto cabe recordar que algunos de los principios del movimiento agilista son que "los usuarios y desarrolladores deben trabajar juntos diariamente durante el proyecto" y que "el método más eficiente y eficaz de compartir la información es la conversación cara a cara". Lo que facilita la definición de las características del sistema a desarrollar, como también la difusión de este conocimiento al equipo de desarrollo dado el estrecho y frecuente contacto con los usuarios. Estableciéndose una diferencia con los proyectos que estructuran los equipos de desarrollo en términos de roles, donde existen analistas, diseñadores y programadores que se dividen las tareas. En este caso, son sólo los analistas quienes se reúnen con los usuarios, generalmente en una sucesión de entrevistas distanciadas temporalmente y donde el objetivo es acordar y documentar las características deseadas.

A través de la colaboración y el diálogo permanente de los participantes, sus modelos mentales y experiencia son compartidos. Es la socialización que plantea Nonaka, que se fundamenta en esta posibilidad de poder compartir el tiempo y espacio haciendo colectivo conocimiento tácito.

En los enfoques tradicionales, donde los requerimientos deben ser especificados previos al desarrollo, el equipo a cargo de tareas de diseño y construcción interactúa poco o nada con los usuarios con el objetivo de entender los requerimientos del sistema. Basando su trabajo fundamentalmente en los documentos de especificación. Este enfoque es adecuado sólo para

sistemas cuyos requerimientos se sabe con certeza permanecerán estables. Pero en los escenarios actuales de rápido y constante cambio los requerimientos son inestables

La práctica de mantener a los desarrolladores en el mismo lugar de trabajo que los usuarios permite a los métodos ágiles establecer una comunicación permanente para "ajustar" los requerimientos a lo largo del ciclo de desarrollo. Una exigencia para lo anterior es la necesidad de co-localización, lo que para algunas organizaciones no es posible. Existen estudios sobre el uso de métodos ágiles en ambientes de trabajo distribuido que muestran cómo el uso de herramientas para GC han posibilitado levantar en parte esta restricción [18,19].

La GC y sus líneas principales de desarrollo: extracción de información, la administración de información/conocimiento y procesamiento de flujos de trabajo, han permitido el surgimiento de herramientas de apoyo. Existen herramientas especialmente diseñadas para estos propósitos (GroupWare, Call Center, Gestión Documental, Portales Corporativos, Herramientas de Mapeo, Mapas de expertos, etc.) y otras de uso más general que permiten apoyar los procesos de GC (Internet/Intranet, Correo electrónico, Business Intelligence, etc.)

Las herramientas pueden clasificarse en [20]:

- De Personalización: las que permiten el acceso de forma automática a la información que ha sido seleccionada anteriormente, sin necesidad de realizar el mismo tipo de búsqueda más de una vez. Estas herramientas ofrecen la obtención de información sobre temas afines en distintos momentos en el tiempo.
- De Trabajo en Grupo: Son herramientas que permiten generar procesos de colaboración, distribuir y sincronizar tareas en la organización con el objetivo de aumentar la eficacia, reduciendo el tiempo para obtener el conocimiento necesitado. Estas tienden a realizar una gestión integral del conocimiento en una institución.
- De Portales Corporativos: Estos portales permiten, el acceso de las personas a contenido personalizado y además ayudan a crear ambientes de colaboración, por lo que también se les conoce como portales del conocimiento. Una característica que los hace muy útiles es que el usuario tiene acceso a una gran cantidad de información, que no necesariamente está almacenada en la organización, sin tener que cambiar de aplicación.
- De Simulación: Estas simulan el esquema de realización de un proyecto complejo y se basan en los procesos del pensamiento humano con la finalidad de rectificar los errores que puedan presentarse al planificar la ejecución del mismo.

Haciendo uso de este tipo de productos los equipos pueden comunicarse y colaborar aún estando dispersos geográficamente. Tampoco existe una limitación de temporalidad, ya que no se exige un tipo de comunicación sincrónica, lo que amplia las posibilidades de conformación de los grupos de trabajo en régimen de turnos y/o diferentes usos horarios.

4.4 Los Individuos y el Ambiente de Trabajo

Conocer "quien sabe qué" en una organización o equipo de trabajo puede ser muy valioso para el éxito de un proyecto. Los procesos ágiles para hacer "visible" el conocimiento de un miembro del equipo a los demás realizan reuniones periódicas (diarias o semanales), donde cada miembro informa lo realizado desde la última reunión. Entonces, cada miembro sabrá a quien contactar cuando se requiera. Lo más cercano a esto que ofrecen los métodos tradicionales es acudir a quienes firman los documentos..

El manifiesto agilista dice: “Construya los proyectos alrededor de individuos motivados. Déles el ambiente y apóyelos en lo que necesiten. Y confíe en ellos”. Lo que hace mucho sentido dada la naturaleza social del proceso de desarrollo. Es importante desarrollar las confianzas necesarias entre los miembros, usuarios y desarrolladores. Lo que facilitará la creación, compartición y difusión del conocimiento [21]. La propiedad colectiva del código, reuniones periódicas, usuarios co-localizados, programación en parejas, son prácticas que requieren de confianza para poder ejecutarse, y que a su vez promueven la confianza. Un factor relevante en la socialización, tiene relación con los contactos y colaboraciones voluntarias de los miembros del equipo.

En el enfoque agilista se considera equipos de trabajo polifuncionales, en cambio los proyectos tradicionales trabajan con equipos subdivididos en roles. Los roles permiten subdividir las tareas, lo que puede facilitar la asignación a diferentes proyectos que se desarrollan en paralelo. Así, se podrá optimizar el uso del recurso “analista”, “diseñador” o “programador” a similitud de un proceso de fabricación. Una desventaja de este enfoque es la falta de comunicación y flujos de información entre los diferentes roles de un equipo asignado a un proyecto, lo que dificulta la socialización (el compartir conocimiento tácito), perpetuando el conocimiento aislado de cada miembro.

Las herramientas de comunicación entre los diferentes roles son los productos intermedios de cada una de las fases sobre las que tienen responsabilidad. El conocimiento de un rol que se debe entregar a otro, debe ser externalizado y documentado. Así, el analista elaborará la especificación de requerimientos que será utilizada por el diseñador quien elaborará la especificación de diseño para el programador. Equipos polifuncionales pueden facilitar la transferencia de conocimiento. Más aún, cuando existe incertidumbre respecto a los requerimientos y el entorno es muy dinámico.

5. Conclusiones

Los enfoques de desarrollo Tayloriano se caracterizan por compartir conocimiento explícito, externalizado en documentos o repositorios de información. Intentando reutilizar la experiencia ganada en proyectos anteriores definiendo las estructuras necesarias para soportar el aprendizaje colectivo, y asegurar la madurez y los procesos de mejoramiento de sus procesos de desarrollo. A la luz de lo planteado por Nonaka, estos enfoques no consideran instrumentos que permitan abordar cómo los involucrados internalizan conocimiento explícito o puedan compartir conocimiento tácito que no es externalizado.

Los métodos ágiles se fundamentan en la socialización, por medio de la comunicación cara a cara, la colaboración, y la programación en parejas, para compartir conocimiento tácito entre los desarrolladores y usuarios. Cuando externalización e internalización son usados para transferir conocimiento, los métodos ágiles también se basan en la comunicación y colaboración. Considerando a los usuarios un grado de participación relevante, directa y continua en la elicitación de requerimientos y acercamiento a la problemática del dominio. El enfoque iterativo de desarrollo utilizado posibilita una buena retroalimentación y un aprendizaje continuo entre los usuarios y desarrolladores. Para facilitar el aprendizaje entre los desarrolladores, los métodos ágiles hacen uso de diversas prácticas, entre las que se destacan: reuniones periódicas (diarias o semanales), programación en parejas, propiedad colectiva del software, refactorización.

Los métodos ágiles privilegian las personas, la comunicación y la colaboración. Lo que facilita el compartir conocimiento tácito. Fomentan también, una cultura de confianza y motivación, lo que a su vez posibilita la implementación de herramientas para compartir conocimiento explícito.

Ciertos dominios pueden ser más adecuados a un tipo de proceso. Y en general, algunos aspectos del desarrollo de software se beneficiarán del enfoque agilista mientras otros obtendrán beneficios de un enfoque tradicional (predictivo, menos ágil o Tayloriano). Desde esta perspectiva los procesos de desarrollo de software pueden ser clasificados dentro de un amplio espectro dependiendo de su "grado de agilidad". Lo importante es saber ubicarse debidamente dentro de él y optar por el tipo de proceso y herramientas que mejor sirvan a cada proyecto.

En los extremos de este espectro hay dos escuelas de pensamiento en el desarrollo de software. Una de ellas mueve a los desarrolladores a respetar un acabado plan. Y se traduce en que a partir de un diseño detallado se pueda construir el código que se espera, de acuerdo a unos requerimientos muy bien documentados. La otra escuela de pensamiento sostiene que es mejor enfrentar el desarrollo en pequeñas iteraciones.

La primera deja poco espacio para la generación de conocimiento a través de la experimentación, y se sostiene en la deliberación y revisión. El objetivo debe ser balancear experimentación y revisión. Este trade off debe ser resuelto en base a los costos involucrados en la generación del conocimiento. Si el costo de probar es muy alto, es preferible generar conocimiento a través de la deliberación y revisión. En otros casos, la experimentación puede ser menos costosa y permitir generar conocimiento rápidamente.

Una combinación de experimentación, revisión e iteración en muchos casos puede presentar buenos resultados. Muchos autores a la fecha vienen indicando que las metodologías tradicionales no son totalmente adecuadas para todos los desarrollos software. Las razones son diversas, las principales son la falta de flexibilidad de los procesos de desarrollo frente a cambios y su excesiva documentación. Como se ha planteado en este trabajo los métodos ágiles y los tradicionales no son estrictamente competidores directos. Cada uno de ellos tiene su propio segmento de aplicación o terreno. Y pueden ser usados en proyectos con diferentes características: los métodos tradicionales son más adecuados en grandes proyectos con requerimientos estables, aplicaciones críticas y donde la memoria corporativa tiene relevancia. Los métodos ágiles en cambio se adecuan mejor en ambientes dinámicos, con equipos de trabajo pequeños y produciendo aplicaciones no críticas. También son una buena elección cuando se trabaja con requerimientos desconocidos o inestables, garantizando un menor riesgo ante la posibilidad de cambio en los requerimientos.

6. Referencias

[1] K. Beck et al, *Manifesto for Agile Software Development,* http://agilemanifesto.org/

[2] J. Ridderstrale & K. Nordstrom, *Funky Business: Talent Makes Capital Dance*, Financial Times / Prentice Hall, EEUU, 2000

[3] T. Chau y F. Maurer, *Knowledge Sharing un Agile Software Teams*, http://sern.ucalgary.ca/~milos/papers/2004/ChauMaurer2004.pdf/

[4] F. Taylor, *The Principles of Scientific Management*, Dover Pubns, 1998

[5] E. Bueno, *La Gestión del Conocimiento nuevos perfiles profesionales*, Junio 1999. http://www.sedic.es/bueno.pdf

[6] H. Koonz y H. Weihrich, *Administración: una perspectiva global*, McGraw Hill, España, 1995

[7] S. Nonaka y N. Takeuchi, *La Organización creadora de Conocimiento*, Oxford University, 1995

[8] Instituto Universitario Euroforum Escorial, *Medición del Capital Intelectual: Modelo Intelect*, Madrid, 1998

[9] G. Mentzas, *The two faces of Knowledge Management*, http://imu.iccs.ntua.gr/Papers/O37-icg.pdf/

[10] D. Apostolou y G. Mentzas, *Experiences from KM implementations in companies of the software sector*, Business Process Management, Vol 9 No 3, 2003, MCB UP Limited

[11] U. Kelter et al, *Do we Need 'Agile' Software Development Tools?*, http://pi.informatik.uni-siegen.de/

[12] M. Fowler, *The New Methodology*, http://www.programacionextrema.org/articulos/newMethodology.html/

[13] J. Bozo y B. Crawford, *Métodos Ágiles como Alternativa al Proceso de Desarrollo Web*, IX Congreso Argentino de Ciencias de la Computación CACIC 2003, La Plata, del 6 al 10 de Octubre 2003

[14] W. H. Inmon, *Choosing the right approach to DW: Big Bang vs. Iterative*, www.billinmon.com

[15] B. Meyer, *Software Enginnering on Internet Time*, IEEE Computer, Vol. 34, Nº35, May 2001

[16] M. Poppendieck y T. Poppendieck, *Lean Development Thinking Tools for Agile Software Development Leaders*, Addison-Wesley Pub Co, 2003

[17] M. J. Escalona y N. Koch, *Ingeniería de Requisitos en Aplicaciones para la Web: Un estudio comparativo*, www.pst.informatik.uni-muenchen.de/personen/kochn/ideas03-escalona-koch.pdf

[18] P. Baheti, L. Williams, E. Gehringer, y D.Stotts, *Exploring Pair Programming in Distributed Object-Oriented Team Projects*, In Proceedings of XP/Agile Universe 2002, Chicago, August 4-7, 2002, Lecture Notes in Computer Science 2418, Springer Verlag

[19] I. Rus, M. Lindvall, *Knowledge Management in Software Engineering*, IEEE Software, May-June 2002

[20] B. Crawford, J. Bozo y K. Rojas, *Marco teórico para la proposición fundamentada de una herramienta computacional para la Gestión de Competencias*, XI Encuentro Chileno de Computación, Jornadas Chilenas de Computación 2003, Chillán, del 3 al 7 de Noviembre 2003

[21] A. Cockburn y J. Highsmith, *Agile Software Development: The People Factor*, IEEE Computer, Vol. 34, No.11, 2001