Experiencias con Herramientas Colaborativas para la Gestión del Conocimiento en Entornos Distribuidos

Karla Mendes Calo[1] Karina Cenci[1] Pablo Fillottrani[1,2]

[1]Laboratorio de Investigación en Ingeniería de Software y Sistemas de Información
Departamento de Ciencias e Ingeniería de la Computación
Universidad Nacional del Sur
[2]Comisión de Investigaciones Científicas, Provincia de Buenos Aires
{kmca,kmc,prf}@cs.uns.edu.ar

Resumen El desarrollo global del software es una tendencia en crecimiento, por el auge de la globalización y fusión de empresas. Como en cualquier desarrollo de un proyecto la meta es alcanzar un producto de calidad y que la gestión del proceso sea eficiente, visible y consistente. Además, la construcción de productos requiere el uso de conocimiento. Otro aspecto significativo es la gestión del conocimiento aplicada al proceso de desarrollo, con especial énfasis, en los casos que los requerimientos son cambiantes.
La importancia de la administración del conocimiento motivó el inicio del estudio y análisis de herramientas colaborativas distribuidas para la mejora en el manejo de proyectos y así alcanzar beneficios en el proceso y por ende en el producto. La relevancia de este trabajo es la presentación de experiencias en el uso de herramientas colaborativas para la gestión del conocimiento para el desarrollo global y distribuido de software. Y la comparación de las mismas, que servirá como parámetro al momento de la elección de una herramienta para el desarrollo de un proyecto distribuido.

Palabras claves: Gestión Conocimiento - Desarrollo Global de Software - Procesos Ágiles

1. Introducción

La disciplina de ingeniería de software promueve la utilización de principios, metodologías y herramientas basadas en formalismo y rigurosidad, abstracción, modularidad. Y como concepto importante, la consideración del software como un producto, en el cual es importante verificar la calidad del mismo y garantizar los requerimientos especificados. Además debemos tener en cuenta que para la producción de software, se requiere de un uso intensivo del conocimiento.

En la actualidad y con gran auge, como influencia de la globalización, las fusiones de algunas empresas y entornos altamente cambiantes, se han comenzado a desarrollar proyectos de desarrollo con participantes que están trabajando en diferentes regiones

de una misma organización. Asimismo, las metodologías tradicionales de desarrollo de software llevan asociado un marcado acento en el control del proceso, promoviendo la producción de artefactos. Estos procesos, denominados tradicionales, han demostrado ser efectivos en proyectos de gran tamaño, particularmente en lo que respecta a la administración de recursos a utilizar y a la planificación de los tiempos de desarrollo. Sin embargo, el enfoque propuesto por estos métodos no resulta el más adecuado para el desarrollo de proyectos donde los requerimientos del sistema son muy cambiantes, surge la necesidad de reducir drásticamente los tiempos de desarrollo y al mismo tiempo producir un producto de alta calidad.

Como alternativa a los métodos tradicionales de desarrollo, surgen las Metodologías Ágiles. Éstas mantienen las prácticas esenciales de las metodologías tradicionales, pero se centran en otras dimensiones del proyecto, como por ejemplo: la colaboración con los usuarios durante todas las etapas del proceso de desarrollo, y el desarrollo incremental del software con iteraciones muy cortas que entregan una solución a medida. Las Metodologías Ágiles son más adaptativas que predictivas, y más orientadas a las personas que a los procesos. Es por ello que las organizaciones que utilizan estas metodologías, se han visto obligadas a cambiar su filosofía y plan de trabajo para mantener y convertir dicha información en conocimiento, y que la misma sea accesible para todos los integrantes del equipo de desarrollo de tal forma que permita aplicar, adaptar e integrar cambios rápidamente.

En la sección 2 se presentan un conjuntos de conceptos utilizados (proceso ágil, gestión del conocimiento y desarrollo global del software). Sección 3 presenta las características de la gestión del conocimiento aplicada al desarrollo global y distribuido. Sección 4 introduce varias experiencia de uso de herramientas colaborativas distribuidas para la gestión del conocimiento en el desarrollo de proyectos aplicando la metodología ágil Scrum y una comparación de las funcionalidades que brindan. Y por último, en la sección 5 se presentan las conclusiones.

2. Conceptos Previos

Proceso Ágil El Manifiesto Ágil [2] es un documento que incluye cuatro postulados y una serie de principios asociados para el desarrollo ágil. Sus postulados son: 1) Valorar al individuo y a las interacciones del equipo de desarrollo por encima del proceso y las herramientas. 2) Valorar el desarrollo de software que funcione por sobre una documentación exhaustiva. 3) Valorar la colaboración con el cliente por sobre la negociación contractual. 4) Valorar la respuesta al cambio por sobre el seguimiento de un plan.

En función de estos postulados, el desarrollo ágil de software, combina una filosofía y un conjunto de directrices de crecimiento; en donde la filosofía busca la satisfacción del usuario final junto a la entrega temprana o rápida del mismo; la formación de equipos de proyecto con alta motivación, es decir, trabajo colaborativo, métodos informales de programación y simplicidad general del desarrollo del software. Todo esto para alcanzar las metas propuestas por los equipos de trabajos de una manera más rápida y eficaz con menos desgate tanto humano como temporal. El ciclo de desarrollo que aplican estas Metodologías Ágiles es iterativo e incremental. Permite entregar el software en partes pequeñas y utilizables, conocidas como incrementos.

Gestión del Conocimiento. El conocimiento requiere entender la información. No está contenido solamente en información, sino en las relaciones entre los ítems de

información, su clasificación y metadata. La experiencia es aplicada al conocimiento. La gestión del conocimiento requiere conocer los niveles del conocimiento, características y las fases del ciclo de evolución para su implementación [10].

El conocimiento puede ser *explícito* ó *tácito*. El *explícito* en general es más fácil de manejar, ya que es un conocimiento codificado y reconocido, como puede ser la documentación de procesos. El *tácito* es un conocimiento personal que los empleados alcanzan a través de la experiencia; este tipo de conocimiento es más difícil de comunicar y está influenciado por la integralidad de la persona. El alcance del conocimiento puede involucar una persona, un grupo, una organización, múltiples organizaciones.

El ciclo de evolución del conocimiento define las fases del conocimiento organizacional. Las fases son:

1.- *Creación*. Los miembros de la organización desarrollan conocimiento a través del aprendizaje, resolviendo problemas, innovación, creatividad, e incorporación de fuentes externas.

2.- *Captura*. Los miembros adquieren y capturan información sobre el conocimiento en forma explícita.

3.- *Organización*. Las organizaciones organizan, transforman ó incluyen conocimiento en material escrito y bases de conocimiento.

4.- *Acceso*. Las organizaciones distribuyen el conocimiento por medio de la educación, programas de entrenamiento, sistemas automatizados basados en conocimiento, o redes expertas.

5.- *Uso*. Es la última fase, en donde la organización utiliza el conocimiento, esto es, el conocimiento está disponible cuando sea que se lo necesite.

Desarrollo Global de Software El término *desarrollo global de software* (DGS) ha sido utilizado por diferentes autores. Se han propuesto varias definiciones para este término. Definición 1 [9,11]: *Las actividades de software globalmente distribuidas entre varios sitios que están ubicados en diferentes países y continentes.* Definición 2 [6]: *Todas las actividades del ciclo de vida del software donde el proyecto involucre actores que están dispersos al menos en dos sitios los cuales están separados por límites fronterizos de países o continentes, y típicamente en diferentes zonas horarias con un grado de distancia socio-cultural entre los actores.*

El DGS puede considerarse que surge como un efecto colateral de la fusión y adquisición de las compañías. Además se lo ha promovido como un modelo beneficioso para el desarrollo de su propio software. Algunas ventajas son: -acceso a empleados rentables, -acceso a gran cantidad de empleados cualificado, -proximidad al mercado y a los clientes, -eficacia del huso horario e -innovación y compartición de mejores prácticas.

El desarrollo de software requiere los procesos de coordinación, control y comunicación. El DGS es un desarrollo de software en un contexto distribuido, hereda los mismos procesos y además requiere considerar las distancias geográfica, temporal y socio-cultural.

La distancia en DGS puede potenciar algunos efectos negativos, como es la dificultad en la comunicación, la cantidad de horas disponibles para una comunicación sincrónica, el costo para los viajes [3]. Pero también hay que considerar las oportunidades que ofrece la naturaleza del DGS. Algunos aspectos del trabajo distribuido permiten maximizar las horas de desarrollo de un día, acercarse a las necesidades de los clientes y del mismo mercado, promover la documentación.

3. Gestión del Conocimiento en el Desarrollo Distribuido y Global del Software en el Proceso Ágil

«*Como las organizaciones incrementan sus esfuerzos en el desarrollo global de software, deben desarrollar nuevos métodos y modelos para el manejo de la enorme cantidad de conocimientos que participan en estos proyectos.*» [4]

La tendencia actual en las organizaciones está en promover el uso intensivo del conocimiento en las iniciativas de software global y distribuido. El uso intensivo del conocimiento en el desarrollo gobal de software plantea interesantes desafíos para las organizaciones. El conocimiento debe ser manejado en todas las etapas de desarrollo del software (desde la captura de los requerimientos, el diseño, la creación de programas y pruebas, la instalación del software y el mantenimiento) e incluso se extiende para mejorar los procesos de planificación del desarrollo de software.

En ambientes distribuidos, los conocimientos necesarios (expertos, mejores prácticas, ideas, y así sucesivamente), se propaga a través de localidades y (en casos como la contratación externa) a través de organizaciones. Por lo tanto, coordinar y sintetizar el conocimiento es un desafío. En un estudio realizado por Desouza y otros [5], un gerente en el Laboratorio de Motorola remarcó: «*Cometemos errores, y sabemos que estos errores se hicieron en el pasado, sin embargo, constantemente reutilizamos los errores.*»

Desouza y otros [4] proponen los siguientes pasos como parte del ciclo para implementar la gestión del conocimiento en el DGS: seleccionar la estrategia, crear el sistema de gestión, reutilizar el conocimiento y capturar el conocimiento. El paso de selección de la estrategia está asociado con la forma de gestionar el conocimiento: *centralizada*, *central-regional* ó *regional-local*. La toma de decisión para la gestión está basada en algunos factores como: naturaleza del trabajo, el grado de autonomía de las oficinas, los recursos disponibles. Después de elegida la estrategia, el próximo paso es poner en marcha el proceso de gestión del conocimiento. Para el desarrollo del proceso se definen modelos arquitectónico y de implementación. Las arquitecturas que actualmente se utilizan son: modelo *cliente-servidor*, modelo *peer-to-peer* y modelo *híbrido*. Para la implementación también se han identificado tres modelos primarios: modelo *centralizado*, modelo de *servicios-compartidos* y modelo *observador*. La elección del modelo de implementación está altamente relacionado con el grado de madurez de la organización en el desarrollo de software y la variabilidad de trabajos de software en la organización. En una organización que tiene un alto grado de madurez, en general la opción más común es que el departamento de tecnología opte por una solución global. En los casos en que el grado de madurez es menor, o en organizaciones que tienen una alta variabilidad de trabajos de software puede ser conveniente elegir otra posición como la de consultor o asesor, esto es seleccionando el modelo de servicios-compartidos u observador.

La propuesta de Avram [1] para prácticas de DGS, a diferencia de la Desouza, centra su trabajo en las praćticas sociales y no en cuestiones técnicas. Esto es, qué hacen las presonas en sus prácticas de trabajo, más bien que en lo que conocen.

La gestión del conocimiento en el desarrollo distribuido se logra brindando a los desarrolladores una forma fácil y eficiente de encontrar la información que necesitan para el desarrollo. Facilitar al grupo de desarrollo una manera en la que pueda hacer seguimiento de los objetivos planteados al comienzo del proceso de desarrollo. Registrar los cambios que han surgido y las implicaciones que estos cambios han forzado. Apoyar a los desarrolladores, donde cada uno se pueda retroalimentar y construir sobre lo

que ya se ha hecho y sobre retroalimentaciones de otros miembros del grupo. Brindar apoyo en términos de colaboración entre los diferentes miembros del grupo para que complementen su conocimiento y construyan sobre los resultados producidos por la interacción entre los miembros del equipo.

Una de las razones principales por el cual las metodologías ágiles reducen el énfasis en la documentación y sí se enfatiza en la documentación sobre el código fuente, es porque es muy costoso para la organización crear y particularmente mantener al día los requerimientos del cliente que van cambiando a medida que se avanza en cada iteración, para llevar a cabo la construcción de la aplicación. Para cualquier proyecto de desarrollo de software, especialmente utilizando metodologías ágiles, hay una serie de información que contienen conocimientos útiles que no necesita ser mantenido activamente por el equipo de desarrollo. Esta información puede estar presente en e-mails entre el cliente y miembros de equipo, posteos acerca de discusiones técnicas con alternativas para su resolución, cuestiones técnicas a tener en cuenta, lecciones aprendidas, historias gestionadas por un grupo de procesos centrales. Por lo tanto, este tipo de información disponible para el equipo de desarrollo, puede ayudarlos a llevar a cabo con éxito sus tareas [7] [8]. Dado que el proceso de construcción de software utilizando Metodologías Ágiles es un proceso cíclico e iterativo, hay que tener en cuenta que la gestión de este conocimiento que se ha identificado anteriormente, implica además asegurar la distribución de este conocimiento entre los miembros del equipo, y combinarlo además con nuevo conocimiento que se irá sumando a través de las iteraciones durante el proceso de desarrollo.

Los proyectos de software se caracterizan por su presupuesto y cronograma, y en general suele sueceder que consistentemente se sobrepasan los mismos (tiempo y coste). Para mejorar estas prácticas es importante contar con el conocimiento del análisis post hoc de las tareas y la derivación de las lecciones aprendidas.

Un aspecto importante a tener en cuenta, es que independientemente de la o las herramientas que se utilicen para gestionar el conocimiento, es necesario mantener la trazabilidad de los artefactos que se creen a lo largo de las iteraciones hasta la finalización del proyecto. Esto significa por ejemplo que, a partir de una línea de código, se puede determinar a qué número de requerimiento del usuario corresponde, y de allí poder acceder a todos los artefactos que nos permitan seguir la traza de todo lo que se ha realizado, problemas con los que se han encontrado y sus posibles soluciones, tests ejecutados, resultados de los tests, defectos asociados a los requerimientos, y el estado actual de su resolución.

4. Herramientas para Gestionar el DGS

Los casos de estudio presentados corresponden a una organización dedicada al desarrollo ágil de software, en el que un cliente contrata a un equipo para que se le construya una aplicación a medida. Para los proyectos se utiliza un desarrollo distribuido, ya que el equipo está formado por personas que trabajan en diferentes locaciones y además el cliente reside en otra locación. En las experiencias presentadas, además de requerir los procesos de coordinación, control y comunicación, se le agregan las distancias socio-cultural, temporal y geográfica. La metodología ágil utilizadas en todos los casos es Scrum. Esta metodología se aplica para proyectos con alto ratio de cambio de requerimientos. Su principal característica es la definición de sprints, cada una de las iteraciones del proceso con una duración máxima de 30 días. El resultado de cada

sprint es un incremento ejecutable que se muestra al cliente. Otra característica importante son las reuniones diarias que se llevan a cabo a lo largo del proyecto. Dichas reuniones no requieren más de 15 minutos del equipo de desarrollo y su objetivo son la coordinación e integración del producto a entregar.

4.1. Experiencia con proyectos utilizando SharePoint

Para llevar adelante el proyecto de desarrollo de software distribuido era necesaria una herramienta que permitiera la colaboración y la comunicación en todo el proceso. Uno de los puntos sobre los que se necesitaba profundizar y promover era la gestión del conocimiento, de manera que, independientemente de los roles de cada integrante del equipo, la locación y su huso horario, se requería contar con la posibilidad de que todos accedieran a la información completa.

Office SharePoint Server 2007, es una herramienta colaborativa integrada al correo electrónico y a los navegadores web y que simplifica la interacción con el contenido, gestionando adecuadamente toda la información, facilitando no solo el acceso sino la búsqueda de la misma.

Una vez identificados los artefactos necesarios, se crearon las plantillas para cada uno de ellos: por ejemplo Historias de Usuario, Test de Aceptación, Wikies, entre otros. La organización en la herramienta se llevó a cabo a través de un repositorio en el que se describían las historias de usuario cada una en una wiki asociada. Se creó una carpeta para mantener el Product Backlog, el cual contiene los requerimientos funcionales y no funcionales.

El Product Backlog permite la navegación entre los artefactos, y mantener la trazabilidad entre los mismos, pudiendo acceder a cualquier parte en la documentación. A partir de una Historia de Usuario, podemos conocer a qué Sprint pertenece, acceder a la información de la Wiki, conocer quiénes colaboraron en la creación, modificación y mantenimiento de la información actualizada de dicho artefacto, quiénes fueron los encargados de realizar cada cambio y con la posibilidad de acceder al detalle del perfil. Además, facilita el acceso a los Tests de Aceptación, chequear el estado de los mismos, si fueron ejecutados o no, por quién, resultados de las corridas, al igual que la información del perfil que intervino. Dar a conocer al resto del equipo, soluciones a diferentes problemas que se presentan durante el avance del proyecto, pudiendo ser de utilidad tanto para los integrantes de este equipo, como a integrantes de otros equipos.

4.2. Experiencia con proyectos utilizando Team Fundation Server (TFS)

La herramienta TFS presenta las mismas funcionalidades que SharePoint e incorpora el manejo de gestión de defectos, el uso de control de versiones, gestión de cambios, asignación de tareas a los integrantes del equipo, y la gestión de reportes. Se cuenta con información que ayuda a la toma de decisiones posibilitando realizar cambios correctivos, ante la visibilidad completa del estado del proyecto. Todo integrado en la misma herramienta colaborativa.

Esta herramienta favoreció la capacidad de seguimiento de tareas, visibilidad para los integrantes del equipo en tiempo real asegurando la calidad del producto, manteniendo la trazabilidad entre los artefactos. A partir de una Historia de Usuario, se conoce el Sprint al que pertenece, se accede a la información de la Wiki, se conoce quiénes colaboraron en la creación, modificación y mantenimiento de la información

actualizada de dicho artefacto, el tiempo que insumió llevar a cabo cada tarea en función de la complejidad de la misma. Se accede también a los Tests de Aceptación, resultados de las ejecuciones, al igual que la información del perfil, y los tiempos insumidos en cada una de estas tareas. La gestión de los defectos se lleva a cabo en la misma herramienta colaborativa. Los defectos cuentan con su propio ciclo de vida, prioridad, impacto, pudiéndose obtener métricas, utilizadas luego como base de información en los sprints siguientes, y como información para oportunidades de mejora para otros proyectos.

4.3. Experiencia con proyectos utilizando Jira

Otra herramienta utilizada para la gestión de conocimiento en proyectos de software para metodologías ágiles es Jira. Esta herramienta presenta funcionalidades similares a las presentadas por TFS. Permite el manejo de control de versiones, notificaciones a los integrantes del equipo de nuevas tareas a llevar a cabo o modificaciones en las ya existentes. Al igual que otras herramientas dedicadas a la gestión de conocimiento, registra las operaciones para mantener la trazabilidad entre los artefactos. Cada uno de ellos soporta un ciclo de vida en el que se pueda agregar, quitar o cambiar transiciones si ello fuera requerido. Esta herramienta cuenta con la ventaja adicional que, si la cantidad de usuarios es hasta 10, el costo de la licencia es ínfimo.

Citamos como ejemplo un proyecto iniciado con la herramienta TFS, que luego de un año de la utilización exitosa por parte del equipo de desarrollo, el cliente decidió dejar de usar esta herramienta por una cuestión de costo. Para ello se decidió buscar una herramienta de gestión de información que permitiera importar toda la información ya gestionada. Se eligió Jira ya que cumplía todos los requisitos, esto es la importación de toda la información y la reducción de costos.

4.4. Comparaciones

En el cuadro 1 se presenta una comparación entre las herramientas utilizadas entre las experiencias de gestión del conocimiento en el desarrollo ágil y distribuido para la construcción de un producto de calidad. La meta en todos los proyectos era contar con una única herramienta que permitiera realizar la gestión de conocimiento en el desarrollo de software distribuido y global. En el caso de uso de SharePoint, no se alcanzaron los resultados esperados. Si bien fue una experiencia enriquecedora, de mantenimiento relativamente simple y escalable al comienzo, al incrementar el flujo de tareas comenzaron las dificultades para alcanzar la trazabilidad entre los artefactos. Además, al momento de obtener respuestas a consultas específicas para contar con métricas del proyecto, realizar evaluaciones, y en función de ellas tomar las acciones que hicieran falta ante posibles desvíos que pudieran presentarse, se debía gestionar de manera separada las herramientas para obtener informes útiles para la toma de decisiones.

En cuanto a TFS integra en la misma herramienta el manejo de gestión de defectos, el uso de control de versiones, gestión de cambios, asignación de tareas a los integrantes del equipo, y la gestión de reportes. La herramienta provee información que ayuda a la toma de decisiones posibilitando la realización de cambios correctivos, ante la visibilidad completa del estado del proyecto. Admite la implementación de metodologías de procesos ágiles, a través de las plantillas de Scrum provistas por la herramienta. Con esto, se fortaleció la capacidad de seguimiento total, visibilidad para

	Herramientas		
Atributo comparado	Sharepoint	TFS	JIRA
Escalabilidad	-	√	√
Transparencia de locación	√	√	√
Personalización de plantillas	√	√	√
Ciclo de vida para cada artefacto	√	√	√
Ciclo de vida entre Plantillas	-	√	√
Gestión de Errores integrado	-	√	√
Trazabilidad entre los artefactos	-	√	√
Generación de reportes	√	√	√
Alerta de cambios a integrantes del equipo personalizados	√	√	√
Clasificación por grupo funcional	-	√	√
Gestión de recursos	-	√	√
Seguimiento de tareas	-	√	√
Gestión de control de cambios	-	√	√
Obtención de métricas	-	√	√

Cuadro 1. Tabla Comparativa de Herramientas de Gestión

todos los integrantes del equipo en tiempo real para asegurar la calidad del producto a desarrollar, mantener la trazabilidad entre todos los artefactos generados sin perder la granularidad, y realizar seguimientos de tareas al nivel de detalle que se desee.

En cuanto a Jira presenta similares características funcionales que TSF, con la ventaja de una reducción considerable en el costo de las licencias de uso. La misma depende de la cantidad de usuarios que utilicen la herramienta para un proyecto.

La elección de la estrategia, herramienta, arquitectura e implementación para la gestión del conocimiento dependerá del tipo de proyecto a realizar y de las restricciones que imponga el cliente. En la última experiencia presentada, el que motivó la búsqueda y elección de una nueva herramienta fue el cliente.

En este trabajo, se utilizaron en la evaluación sólo estas tres herramientas. En el mercado existe una gran variedad de herramientas para la gestión del conocimiento que se pueden clasificar de acuerdo a las funcionalidades, como por ejemplo, Google docs, Google group, Collabdev, Knowledge Tree para la gestión colaborativa del conocimiento; Mercurial, Subversion entre otros para el seguimiento de la información.

5. Conclusiones

En la actualidad, hay un conjunto de clientes que eligen el modelo distribuido y global aplicando metodologías ágiles para el desarrollo de sus productos, ya que al disponer de medios de comunicación accesibles y disponibles 24x7, consideran importante el acceso a personal rentable y cualificado. El DGS utiliza, al igual que el desarrollo de software, procesos de comunicación, coordinación y control. Estos procesos se van a ver influenciados por las distancias geográfica, temporal y socio-cultural. Para la gestión de estos desarrollos globales distribuidos es importante estimular y favorecer las diferentes formas de comunicación (formal e informal), por medio de documentación escrita, conferencias, chats, mensajería, conversaciones de pasillo, etc. La buena comunicación permite mejorar el control y coordinación del mismo. Estos modelos de proceso ágil

deben ser soportados por herramientas colaborativas que ayuden desde el inicio hasta la finalización de la construcción del producto, y luego como base de conocimiento para nuevos proyectos.

Todo desarrollo que se realice en forma distribuida tiene naturalmente ventajas y desventajas. Alguna de éstas últimas es que al existir varias locaciones de trabajo, el protocolo de los diferentes procesos se complejiza, además que se potencian los puntos de falla. A pesar de las dificultades, el DGS es una tendencia que va en crecimiento, ya que ventajas como reducción en los costes y la posibilidad de maximizar las horas de trabajo durante un día permiten mejorar el proceso de producción.

En los últimos años, se ha logrado mejorar el uso de las herramientas colaborativas. A partir de la experiencia de los equipos de desarrollo en el uso de estas herramientas surge oportunidades de mejora en la obtención del conocimiento. Cada una de las organizaciones debe adoptar la herramienta que mejor se adecúe a su forma de trabajo.

La elección de la o las herramientas colaborativas debe contemplar además otros aspectos de la organización, como la definición de sus procesos, la definición de sus objetivos, su estructura organizacional, el tamaño del proyecto, el número de integrantes del equipo, la distribución de los integrantes, entre otros. Es por ello que debe integrar una visión más amplia que permita formular propuestas íntegras para mejorar la gestión de los proyectos de software, ya que el objetivo es tomar decisiones en función de información completa, atómica y verdadera.

La continuación del trabajo estará orientada a la investigación y análisis de otras herramientas colaborativas, que mejoren las facilidades de la registración del conocimiento, y la recuperación del mismo de manera rápida, eficiente y veraz. Una meta de este estudio es la definición de una propuesta de clasificación de herramientas teniendo en cuenta las funcionalidades provistas y las posibilidades de adaptación a la forma de trabajo del equipo de desarrollo y del cliente.

Referencias

1. G. Avram. Knowledge work practices in global software development. *The Electronic Journal of Knowledge Management, available online at www.ejkm.com*, 5(4):347–356, 2007.
2. K. Beck, J. Grenning, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Highsmith, A. Hunt, R. Jeffries, J. Kern, B. Marick, R. C. Martin, S. Mellor, K. Schwaber, J. Sutherland, and D. Thomas. *Agile Manifesto*, 2001.
3. E. Ó Conchúir. *Global Software Development: A Multiple-Case Study of the Realisation of the Benefits*. PhD thesis, University of Limerick, May 2010.
4. K. Desouza, Y. Awazu, and P. Baloh. Managing knowlegde in global software development efforts: Issues and practices. *IEEE Software*, 2006.
5. K. Desouza, T. Dingsrør, and Y. Awazu. Experiences with conducting project postmortems: Reports versus stories. *Software Process: Improvement and Practice*, 10(2):203–215, 2005.
6. P. Ågerfalk, B. Fitzgerald, H. Holmströ m, B. Lings, B. Lundell, and E. Ó Conchúir. A framework for considering opportunities and threats in distributed software development. In *International Workshop on Distributed Software Development*, pages 47–61, 2005.
7. Harald Holz and Frank Maurer. Knowledge management support for distributed agile software processes. In *LSO*, pages 60–80, 2002.

8. Harald Holz, Heiko Maus, Naoyuki Nomura, and Martin Schaaf. Second workshop on knowledge management for distributed agile processes: Models, techniques, and infrastructure (kmdap 2005). In *Wissensmanagement*, pages 365–366, 2005.
9. A. Mockus and J. D. Herbsleb. Challenges of global software development. In *Proceedings of the Seventh International Software Metrics Symposium (METRICS '01)*, pages 182–184, 2001.
10. I. Rus and M. Lindvall. Knowledge management in software engineering. *IEEE Software*, pages 26–38, 2002.
11. M. A. Vanzin, M. B. Ribeiro, R. Prikladnicki, I. Ceccato, and D. Antunes. Global software processes definition in a distributed environment. In *Proceedings of the 2005 29th Annual IEEE/NASA Software Engineering Workshop (SEW ' 05)*, pages 57–65, 2005.