Un enfoque inteligente para asistir en la planificación de proyectos ágiles

Guillermo Rodríguez, Luis Berdún, Álvaro Soria, Analía Amandi y Macelo Campo
ISISTAN Research Institute (CONICET-UNICEN) Campus Universitario, Paraje Arroyo Seco, B7001BBO, Tandil, Bs. As.
{guillermo.rodriguez; luis.berdun; alvaro.soria; analia.amandi; marcelo.campo}@isistan.unicen.edu.ar

Resumen La gestión de conocimiento ha ganado importante popularidad en la gestión de proyectos de software. Paralelamente, Scrum se ha convertido en el método ágil más utilizado en la industria del software por estar principalmente focalizado en la gestión de proyectos. En este contexto, existen varias herramientas de administración de proyectos que asisten a los equipos de trabajo en visualizar planes y progreso del proyecto. Sin embargo, estas herramientas no ofrecen asistencia en cuanto a la captura de conocimiento y su reutilización para otros proyectos, derivando así en una Amnesia Organizacional. En este trabajo se propone un enfoque inteligente denominado *Intelligent Virtual Scrum* (IVS) que asiste a los equipos de trabajo a lo largo de las distintas etapas del ciclo de vida del proyecto, permitiendo la captura del conocimiento generado y su utilización en el armado de un plan en cada iteración del proyecto. La evaluación con un caso de estudio demostró la viabilidad y eficacia del enfoque.

Palabras claves: Algoritmos de Planning; Agentes Inteligentes; Planificación de Proyectos; Scrum.

1. Introducción

Actualmente, la gestión del conocimiento es fundamental para la innovación y mejora de productos y procesos, para la toma de decisiones ejecutivas y para la renovación y adaptación de las organizaciones [1]. En particular, la gestión de proyectos se ha convertido en una de las principales preocupaciones en la industria del software [2] ya que es necesaria por varias razones legítimas como la planificaciones de campañas de marketing, programación de actividades de entrega de producto, entrenamiento de usuarios internos, entre otros [3-5].

Los métodos ágiles han ganado gran popularidad en la industria del software en los últimos años, siendo Scrum el más utlizado [6]. Scrum apunta a minimizar el impacto de una infeciente y poco precisa gestión del proyecto asegurándose que la funcionalidad más importante sea desarrollada primero. Sin embargo, todavía existe la necesidad de una gestión eficiente y precisa ya que es la base para la planificación de los recursos, priorización de actividades y negociación de contratos [7, 15, 16].

Existen varias herramientas de administración de proyectos para asistir a las organizaciones en visualizar el diseño preliminar del plan, reportar el estado actual del proyecto, identificar caminos críticos, proveer información de costos y la duración

total del proyecto. Sin embargo, la mayoría de estas herramientas fallan a la hora de dar soporte a los procesos de toma de decisiones y no ofrecen asistencia en cuanto a la captura de conocimiento y la planificación de proyectos. Esto último se debe a que la mayoría se dedican al seguimiento de un único proyecto y no se detienen en recopilar experiencias de proyectos anteriores, desaprovechando de esta manera todo el conocimiento previamente adquirido. Normalmente, esta experiencia es sólo adquirida por el equipo de desarrollo y cuando éste es desarmado se lo lleva consigo, perjudicando a la organización [8] y derivando a ésta en lo que se conoce como Amnesia Organizacional (AO).

Para ayudar a preservar conocimiento sobre las tareas y recolectar información relevante de un proyecto, en este trabajo se presenta *Intelligent Virtual Scrum* (IVS), un enfoque inteligente para administrar proyectos. El objetivo del enfoque es optimizar la calidad del plan del proyecto, la cual está determinada por las evaluaciones que recibirá cada miembro por su desempeño en las tareas asignadas luego de la ejecución de dicho plan. IVS se basa en Scrum y es soportado por *Virtual Scrum* [9] con el fin de asistir al equipo de trabajo a lo largo de las distintas etapas del ciclo de vida del proyecto. El enfoque consta de dos etapas: captura del conocimiento y planificación del Sprint. La captura del conocimiento está basada en un sistema Multi-Agente que captura las interacciones del equipo de trabajo con *Virtual Scrum*, aumentando así el conocimiento organizacional y reduciendo el impacto de la AO, mientras que la planificación consiste en un algoritmo de *Planning* que utiliza el conocimiento capturado y propone un Sprint Backlog, el cual puede ser modificado por el equipo y re-evaluado por el algoritmo.

Para evaluar el enfoque se utilizó un caso de estudio artificial cuyos datos fueron tomados de un proyecto real del curso Taller de Ingeniería de Software de la Facultad de Ciencias Exactas (UNICEN, Tandil, Bs. As). A través de una serie de ejecuciones, se observó la eficacia del enfoque para aprender del conocimiento organizacional y recomendar planes que garantizan una eficiente y precisa gestión del proyecto.

El resto del artículo se organiza de la siguiente manera. La sección 2 introduce el enfoque propuesto. La sección 3 describe el caso de estudio utilizado para evaluar el enfoque. Finalmente, la sección 4 presenta las conclusiones y trabajos futuros.

2. *Intelligent Virtual Scrum*

Este trabajo explora la hipótesis que el uso de la Memoria Organizacional (MO) en combinación con algoritmos de *Planning* es una solución adecuada para asistir a un equipo de trabajo en la planificación de proyectos en un contexto ágil. Para corroborar esta hipótesis, se introduce a *Intelligent Virtual Scrum* (IVS) como solución, en la cual la MO cumple el rol de repositorio del conocimiento generado por Agentes Personales que capturan la interacción con *Virtual Scrum*, y a su vez, se utiliza como entrada para el algoritmo de planeamiento. En este contexto, el plan de trabajo resultante refleja las decisiones tomadas por el equipo de trabajo en iteraciones y proyectos previos. La Figura 1 describe el funcionamiento del enfoque en el contexto de Scrum y con el soporte de *Virtual Scrum*. El proceso comienza con la carga de los requeri-

mientos del usuario (*user stories* en la jerga de Scrum) en el Product Backlog, el cual está soportado por el *Virtual Product Backlog* dentro de *Virtual Scrum.*

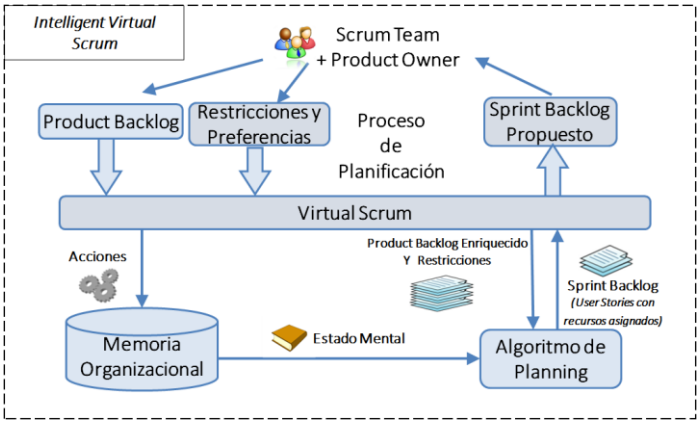

Fig. 1 Interacción entre el usuario y la solución propuesta durante la Planning Session

Una vez definido el Product Backlog, el Scrum Team (responsable de desarrollar los requerimientos del usuario) y el Product Owner (responsable de definir, clarificar y validar los requerimientos del usuario) deben decidir un subconjunto de *user stories* del Product Backlog que serán desarrolladas durante una iteración, denominada Sprint, que puede ir de 2 a 4 semanas. En este punto, el algoritmo de *Planning* toma como entrada el Product Backlog y la MO con el fin de sugerir un conjunto potencial de *user stories* a desarrollar. Como resultado, el algoritmo propone un posible Sprint Backlog que es sometido a aprobación del equipo de trabajo formado por el Scrum Team y el Product Owner. Estos pueden modificar la propuesta del algoritmo e incluso una vez realizadas las modificaciones invocarlo nuevamente a fin de obtener una nueva propuesta. Estas interacciones son observadas por los Agentes Personales e incorporadas como restricciones a la MO.

Una vez planificado y acordado el contenido del Sprint Backlog, el *Scrum Team* está listo para empezar a desarrollar las *user stories* durante el *Sprint*. En un Sprint, cada *user story* sigue un proceso miniatura que consiste en su análisis, diseño, implementación y verificación. En cada paso de este proceso, IVS observa las *user stories* realizadas por cada uno de los miembros del equipo y la valoración recibida por su trabajo, aumentando la información de la M O. Esta información capturada a través de *Virtual Scrum* formará un perfil real del usuario y permitirá conocer en detalle cada una de las habilidades que posee junto con su nivel de trabajo.

Antes de finalizar el Sprint, Scrum propone la realización de dos reuniones: *Sprint Review* y *Sprint Retrospective*. En la *Sprint Review*, el *Scrum Team* presenta el incremento del producto al *Product Owner* por medio de una demo. El resultado de esta reunión es obtener *feedback* y revisar el resultado del Sprint. En este punto el *Scrum Team* y el *Product Owner* registran en *Virtual Scrum* el resultado del análisis, quedando almacenado en la Memoria Organizacional. En la *Sprint Retrospective* el *Scrum Team* reflexiona sobre su performance durante el *Sprint* indicando su desem-

peño para cada *user story*. Esta información es capturada por IVS para formar parte del perfil de cada miembro del equipo y es almacenada en la Memoria Organizacional para ser utilizada en la planificación del próximo *Sprint*. Las siguientes sub-secciones describen las etapas del enfoque: captura del conocimiento y planificación del Sprint.

2.1 Captura del conocimiento

Para la extracción del conocimiento se utiliza el enfoque presentado en [10], en el cual se presenta un sistema de Agentes para la extracción del conocimiento de los usuarios cuando interactúan con una herramienta de administración de proyectos, que en este caso es *Virtual Scrum*. El mecanismo de trabajo se basa en dos tipos de agentes: Agente Personal y Agente Integrador. Cada Agente Personal asiste individualmente a un miembro del equipo de trabajo cuando utiliza *Virtual Scrum* y captura el conocimiento de las acciones que éste realiza. El Agente Integrador se encarga de integrar todo el conocimiento de los Agentes Personales dentro de la base de conocimiento que conforma la Memoria Organizacional.

El conocimiento capturado consiste de la información de las *user stories* realizadas por cada uno de los miembros del equipo durante el Sprint, como así también la valoración que cada miembro recibe por el trabajo realizado durante la *Sprint Retrospective*. Esta información conformará el perfil real del usuario y permitirá conocer en detalle cada una de sus habilidades junto con su nivel de trabajo. La especificación de una *user story* junto con los miembros asignados a la misma permite inferir además de las habilidades reales de cada miembro, cómo éste se desempeña con su equipo de trabajo e incluso con los desafíos de cada *user story*. De esta forma, se puede conocer, dado un miembro de equipo, si éste trabaja bien en *user stories* de complejidad alta, o si trabaja bien junto con un miembro en particular.

Una vez que se captura la interacción de cada uno de los miembros del equipo a lo largo del Sprint y se almacena en forma de transacción, el Agente Integrador utiliza el conjunto de transacciones para extraer patrones de comportamiento de los miembros de equipo. Para extraer dichos patrones, se aplicó un algoritmo de reglas de asociación [11-12], que en este caso puntual es el algoritmo *Apriori* [13]. Por ejemplo, una regla minada puede ser: *habilidad_requerida_us (programacion=Bajo) y habilidad_recurso_humano (programación=Alta)* → *calificacion=(excelente)*. Este patrón en formato de reglas indica que la asignación de un recurso con un nivel de habilidad alto fue asignado a una *user story* que requería nivel de habilidad bajo y obtuvo una calificación excelente. Estos patrones sirven para asistir al equipo de trabajo no sólo en el proceso de estimación de *user stories*, sino también en la asignación de *user stories* a miembros del equipo. Posteriormente, dichos patrones son traducidos para conformar el Estado Mental de IVS, el cual se utilizará en la segunda etapa.

2.2 Planificación del Sprint

Como se mencionó al inicio de la sección, la segunda etapa del enfoque parte de *Product Backlog* y propone un *Sprint Backlog* en base al conocimiento adquirido por

los Agentes. Este *Sprint Backlog* es presentado en forma de plan y se construye a partir de los patrones de comportamiento obtenidos en la etapa anterior.

La Figura 2 describe el flujo de interacción que se produce entre el *Scrum Team* y *Virtual Scrum* para el armado final del *Sprint Backlog*. Se parte de un *Product Backlog* sin planificar creado por el equipo de trabajo donde se especifican las *user stories* del proyecto. Asimismo, se describen restricciones que serán usadas por el algoritmo de *Planning*, las cuales pueden ser características y roles requeridos por una *user story*, impacto sobre el valor agregado establecido por el Product Owner, como así también, las asignaciones iniciales que el equipo considere necesarias. Por ejemplo, una restricción puede ser: *restricción (noPosponerTarea, 10, 100, action(posponerTarea(Tarea))*. Este ejemplo muestra la restricción para que, con un determinado peso y confianza, una tarea no sea pospuesta y que sea planificada para el Sprint actual. Esta información junto con el contenido de la Memoria Organizacional es suministrada al algoritmo que se encarga de armar el primer *Sprint Backlog*. Como algoritmo de *Planning* se utilizó el Algoritmo Ag-Ucpop [14], el cual permite armar una solución al problema de *Planning* pero considerando las preferencias del agente que lo invoca.

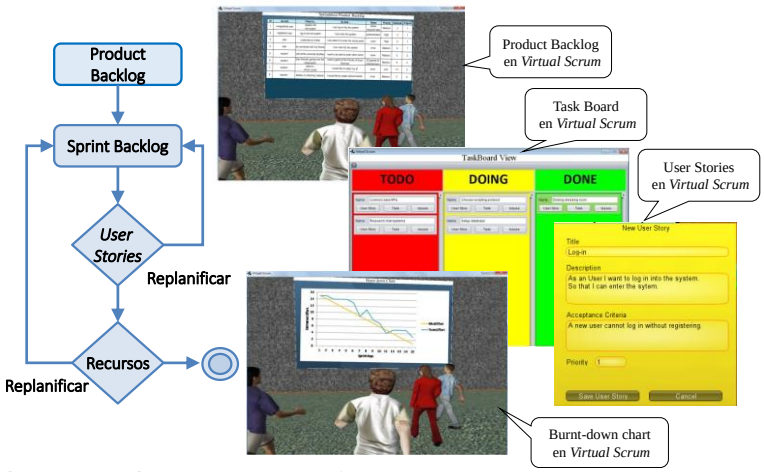

Fig. 2 Edición de la solución propuesta por el enfoque

En este punto, se realiza una conversión que permita llevar un plan de trabajo de la herramienta *Virtual Scrum* a un problema de *planning*. Para dicha conversión, se toma cada una de las *user stories* cargadas en *Virtual Scrum* y son transformadas a diferentes acciones del problema de *planning*. Asimismo, las características de las *user stories* son exportadas como conocimiento que utilizará el algoritmo a la hora de tomar decisiones, los recursos solicitados son transformados a requisitos de la acción, y las asignaciones que realice el equipo son condiciones fijas establecidas en las acciones. Cada uno de estos elementos es utilizado por el algoritmo para alcanzar un plan de proyecto. El estado inicial indica cuáles son los recursos libres y disponibles en el momento de la ejecución del algoritmo y los roles que pueden cumplir cada uno de estos recursos, mientras que el estado final indica el estado en que las tareas deben estar para que el algoritmo de *planning* encuentre una solución satisfactoria.

Una vez que se propone el primer *Sprint Backlog*, el mismo es presentado al equipo, el cual puede introducir los cambios que considere necesarios al plan e incluso eliminar asignaciones realizadas por el algoritmo. Luego de la edición puede solicitar una nueva planificación, invocando nuevamente al algoritmo de planeamiento pero a partir del estado actual e incorporando el nuevo conocimiento que se extrae de la interacción con el equipo. Las características recursivas del algoritmo permiten este procesamiento hasta que el equipo lo considere necesario. Como se ve en la Figura 2, este proceso de edición se puede realizar en dos etapas. Por un lado, se pueden editar las *user stories* del plan decidiendo que una *user story* debería ser pospuesta o viceversa; por otro lado, se pueden editar las asignaciones de los recursos, por ejemplo, decidiendo que el recurso R no debe ser asignado a la *user story*. Como se mencionó anteriormente, cada interacción se traduce en nuevo conocimiento adquirido; es decir, el hecho que el recurso R no sea asignado a la *user story* es traducido a una restricción considerada por el algoritmo de *Planning*.

3. Resultados Experimentales

Para evaluar el enfoque propuesto se utilizó un caso de estudio artificial, cuyos datos fueron recolectados de un proyecto real en el marco del curso Taller de Ingeniería de Software de la Facultad de Ciencias Exactas (UNICEN, Tandil, Bs. As). En el proyecto se especificó un conjunto de *user stories*, las cuales requieren diferentes roles y recursos con determinadas habilidades. Estas *user stories* se establecieron con la misma prioridad y mismo tiempo estimado. Para evaluar el resultado de la solución del proyecto en su totalidad se calcularon 2 métricas del proyecto: la valuación del proyecto que consiste en la sumatoria de las calificaciones de cada una de las asignaciones dividido por la cantidad total de asignaciones; y la duración total del proyecto que consiste en la sumatoria de los tiempos de las *user stories* que se encuentran dentro del camino crítico.

El experimento consistió de 2 fases. En la primera fase se evaluó el enfoque ante situaciones en las que hay recursos muy capacitados que no cumplan satisfactoriamente con el requerimiento al cual fue asignado, o que un recurso con poca experiencia resuelva una *user story* apropiadamente, causando que el enfoque tenga que aprender de situaciones insatisfactorias. El criterio de calificación utilizado establece que los recursos que lógicamente serían los óptimos para cumplir con una *user story* serán calificados negativamente, mientras que por otro lado los recursos asignados más alejados del requerimiento de la *user story* serán calificados positivamente. Por ejemplo, si se requiere un perfil *senior* y se asigna un recurso con perfil *senior* se evalúa negativamente, mientas que si se asigna un perfil *junior* se evalúa positivamente. Lo mismo se realiza con las estimaciones temporales.

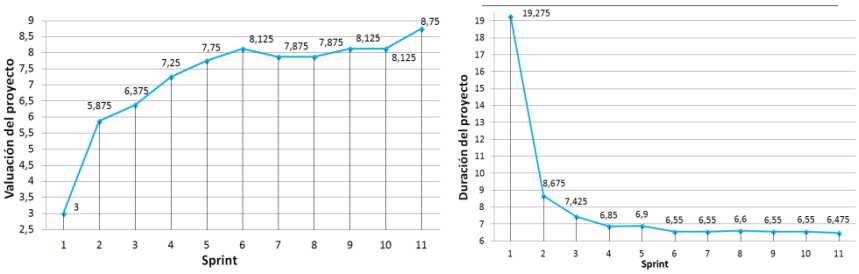

Fig. 3 Resultados de valuación y duración del proyecto

La Figura 3 muestra la evolución de las sugerencias del enfoque a medida que se realizan los distintos Sprints; se puede apreciar cómo el mismo aprende el criterio que hace que se mejore la evaluación y se minimice el tiempo de desarrollo. La segunda fase tuvo como objetivo evaluar la capacidad de recuperación del enfoque, es decir, la capacidad que éste tiene para adaptar el conocimiento almacenado en la MO a un nuevo contexto en el proyecto, el cual está dado por un cambio en la aptitud de los recursos para cumplir satisfactoriamente con los requerimientos asignados. Para esto se cambió la forma de evaluación a la inversa del caso anterior; ahora un *senior* asignado a un lugar de *senior* recibirá una puntuación alta y si se asigna un *junior*, una baja. La Figura 4 muestra cómo la asignación sugerida por la herramienta decae con la nueva forma de evaluación y cómo se va recuperando en los sucesivos Sprints.

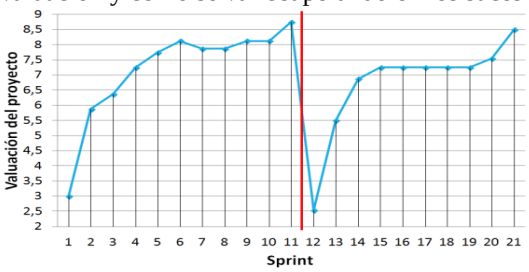

Fig. 4. Recuperación de la valuación del proyecto

4. Conclusiones

En este trabajo se presentó un enfoque inteligente que asiste a un equipo de trabajo en la construcción de un plan de trabajo utilizando el conocimiento adquirido por la organización. Este enfoque interactúa en un contexto de Scrum con una herramienta de manejo de proyectos ágiles conocida como *Virtual Scrum*, y utiliza agentes inteligentes para capturar información generada a partir de la interacción entre el equipo de trabajo y *Virtual Scrum*, la cual se almacena en una Memoria Organizacional. Este conocimiento es utilizado por un algoritmo de *planning* que permite construir iterativamente un plan de proyecto. La principal contribución de este trabajo es que ante un nuevo proyecto el equipo de trabajo dispone de una herramienta que le permite aprovechar cada una de las experiencias ganadas durante los proyectos anteriores. De esta manera, el equipo de trabajo recibe información acorde al mismo aunque sea la primera vez que estén juntos. No obstante, el enfoque presenta algunas limitaciones:

inicialmente no cuenta con una restricción que limite el presupuesto a utilizar, esto se puede subsanar incorporando restricciones que tengan en cuenta el costo del proyecto y los recursos; en segundo lugar, la captura de conocimiento es realizada automáticamente y esto hace que el equipo de trabajo no cuente con una idea exacta del conocimiento; una herramienta de visualización de conocimiento ayudaría a que el equipo de trabajo tenga en mente el nivel de conocimiento capturado antes de planificar el proyecto.

Referencias

[1] Earl, M. (2001) Knowledge Management Strategies: Toward Taxonomy. Journal of Management Information System, 18(1), 215-233.
[2] Kerzner, H. R. (2009) Project Management: A Systems Approach to Planning, Scheduling, and Controlling 10th Ed.
[3] Ireland, L. R. (2006) Project Management: Strategic Design and Implementation.
[4] Cohn, M. (2006) Agile Estimating and Planning.
[5] PMI (2013) A Guide To The Project Management Body Of Knowledge (PMBOK Guides). Project Management Institute, 5th edition. ISBN-13: 978-1935589679.
[6] Schwaber, K. (2002) Agile Software Development with Scrum, Prentice Hall.
[7] Haugen, N. C. (2006) An empirical study of using planning poker for user story estimation. Agile Conference.
[8] Kransdorff, A. (1998) Corporate Amnesia: Keeping Know-How in the Company. Butterworth-Heinemann.
[9] Rodriguez, G., Soria, Á. and Campo, M. (2013), Virtual Scrum: A teaching aid to introduce undergraduate software engineering students to scrum. Comput. Appl. Eng. Educ.. doi: 10.1002/cae.21588
[10] Villaverde, J. (2009) Codificación de la Memoria Organizacional con Perfiles de Usuarios-Tesis Doctorado – Facultad de ciencias Exactas Universidad Nacional del Centro.
[11] Pasquier, N.,Bastide, Y., Taouil, R. and Lakhal, L. (1999) Discovering Frequent Closed Itemsets for Association Rules. In Proceedings of the7th International Conference on Database Theory. Lecture Notes in Computer Science: 398–416. Springer.
[12] Agrawal, R. and Srikant, R. (1994) Fast Algorithms for Mining Association Rules in Large Databases. In Proceedings of 20th International Conference on Very Large Data Bases: 487–499. San Francisco, USA.
[13] Agrawal, R., Imielinski, T. and Swami, A (1993) Mining Association Rules Between Sets of Items in Large Databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data: 207–216, New York, ACM Press.
[14] Berdun, L., Amandi, A. and Campo, M. (2012) An Agent Specific Planning Algorithm. Expert Systems with Applications (39): 4860-4873.
[15] Raith, F., Richter, I., Lindermeier, R. and Klinker, G. (2013) Identification of inaccurate effort estimates in agile software development. In Proceedings of the 20th Asia-Pacific Software Engineering Conference, 67-72, Bangkok.
[16] Trendowicz, A., Münch, J. and Jeffery, R. (2011) State of the Practice in Software Effort Estimation: A survey and literature review. Software Engineering Techniques, LNCS, 4980, 232-245.