Trazabilidad de Procesos Ágiles: un Modelo para la Trazabilidad de Procesos Scrum

Roberto Nazareno[1, 2], Horacio Leone[1, 3], Silvio Gonnet[1,3]

[1] INGAR (CONICET – UTN). Avellaneda 3657, Santa Fe, Argentina
[2] Universidad Nacional de La Rioja, La Rioja, Argentina
[3] Facultad Regional Santa Fe, Universidad Tecnológica Nacional, Santa Fe, Argentina
{rnazareno, hleone, sgonnet}@santafe-conicet.gov.ar

Abstract. La trazabilidad es considerada en metodologías ágiles como un aspecto esencial a incorporar para la producción de software de calidad. Sin embargo, los procesos de desarrollo ágiles en contraposición a los procesos de desarrollo "pesados", no permiten la aplicación directa de las técnicas de trazabilidad tradicionales. En consecuencia, es fundamental desarrollar modelos que permitan trazar los requerimientos bajo el enfoque de los métodos ágiles. En este trabajo se aborda esta problemática centrada en la metodología ágil Scrum. El modelo propuesto es desarrollado con el objetivo de brindar soporte a las siguientes preguntas de competencia: i) ¿Qué eventos originaron un artefacto en particular?; ii) ¿Qué requerimientos guiaron la generación de tal artefacto?; ¿Quiénes son los participantes involucrados en un evento dado? Las respuestas a estar preguntas asistirían las tareas de trazabilidad en proyectos ágiles tales como: Stakeholders con Requerimientos, User Stories con Versiones y Requerimientos con Requerimientos.

Keywords: Scrum, Trazabilidad, Procesos Ágiles

1 Introducción

En los últimos veinte años surgieron diversas propuestas para brindar soporte a la trazabilidad de requerimientos a lo largo del ciclo de vida del sistema. Se entiende por trazabilidad: (i) el grado en el cual se puede establecer una relación entre dos o más productos del proceso de desarrollo, especialmente productos que posean una relación de predecesor-sucesor o superior-subordinado; o (ii) el grado en el cual se puede establecer la razón de la existencia de cada elemento en un proceso de desarrollo de software [1]. Ambas definiciones de trazabilidad explicitan una relación entre artefactos del proceso de desarrollo y su adopción provee un soporte esencial a la producción de software de calidad. En estas dos últimas décadas, las metodologías de desarrollo tradicionales o "pesados" centraron sus prácticas de trazabilidad en el establecimiento de trazas desde requerimientos a otros artefactos de desarrollo. Siendo una traza una relación entre dos o más productos del proceso de desarrollo [1].

La trazabilidad es también considerada en metodologías ágiles como un aspecto fundamental a estudiar para desarrollar sistemas de calidad [2][3]. Sin embargo, los procesos de desarrollo ágiles difieren de los procesos de desarrollo pesados, no permitiendo la aplicación directa de las técnicas de trazabilidad tradicionales en los métodos ágiles.

Los métodos ágiles [4][5] están centrados en el desarrollo, siendo su objetivo proveer una respuesta rápida a los cambios en los requerimientos, a las personas que componen los equipos y a los problemas que surgen durante el proceso de desarrollo [6][7]. En particular, el proceso de ingeniería de requerimientos en métodos ágiles adopta un enfoque de descubrimiento iterativo [8]. El desarrollo ágil ocurre en un ambiente donde la especificación de especificaciones no ambiguas y completas es imposible o incluso no apropiado [2], por lo que frecuentemente no existe un documento con la especificación de los requerimientos a nivel de sistema y de usuario. Sin embargo, muchas organizaciones que producen software empleando métodos ágiles utilizan pruebas para capturar los requerimientos y los mantienen vinculados al código del software [8]. Este escenario no permite aplicar las prácticas de trazabilidad cómo se lo venía aplicando en los métodos pesados. En consecuencia, es fundamental desarrollar modelos que permitan trazar los requerimientos bajo el enfoque de los métodos ágiles. En la actualidad, una de las metodologías de desarrollo de software ágil más utilizada es Scrum [9][10]. Para poder identificar y definir las posibles trazas en la aplicación de Scrum, se propone en este trabajo un modelo conceptual de Scrum. El modelo debe brindar soporte para responder las siguientes preguntas de competencia: i) ¿Qué eventos originaron un artefacto en particular?; ii) ¿Qué requerimientos guiaron la generación de tal artefacto?; iii) ¿Quiénes son los participantes involucrados en un evento dado? Las respuestas a estar preguntas asistirían las tareas de trazabilidad en proyectos ágiles tales como: Stakeholders con Requerimientos, User Stories con Versiones y Requerimientos con Requerimientos. Los beneficios de estas tareas repercuten directamente en: el análisis del impacto de cambios, conformidad en el producto, obediencia del proceso, responsabilidad del proyecto, reproducibilidad de la línea base y aprendizaje organizacional [2].

La siguiente sección presenta el modelo de Scrum propuesto, el cual se organiza a partir de un conjunto de vistas dados por los conceptos eventos, roles y artefactos que guían a Scrum y las relaciones que existen entre esos conceptos. Luego, en la Sección 3 se presenta un caso de estudio, y por último, en la Sección 4, se presentan las conclusiones del trabajo.

2 Modelo para la Trazabilidad de Scrum

Scrum se define como un “framework” basado en los principios ágiles, utilizado para el desarrollo y gestión de productos complejos, como lo son los productos de software. Puede ser visto como un proceso iterativo e incremental que ayuda a involucrar buenas prácticas ingenieriles dentro de una perspectiva iterativa controlada. En las siguientes secciones se presenta el modelo propuesto centrando cada sección en los constructores esenciales de Scrum: *roles*, *eventos*, *artefactos* y las

reglas que permiten asociar estos conceptos (Fig. 1). Luego, se presentan vistas del modelo para brindar soporte a las respuestas de las preguntas de competencia presentadas en la sección previa.

Fig. 1. Conceptos principales de Scrum.

2.1 Roles

Los roles describen las responsabilidades y niveles de autoridad de individuos o grupo de individuos que participan de manera activa en el proceso. En Scrum se definen los roles *Scrum Team*, *Product Owner*, *Scrum Master*, y *Development Team*. La Fig. 2 incluye tales roles y las relaciones entre los mismos. Un proceso Scrum es llevado a cabo por un equipo de trabajo denominado *Scrum Team* (Fig. 2). Este equipo está conformado por diferentes participantes del proceso de desarrollo, provenientes de la organización desarrolladora del software (*ScrumMaster* y el *DevelopmentTeam*) y de la organización que requiere el software (*ProductOwner*).

El *ProductOwner* (Fig. 2) es un individuo (*Individual* en Fig. 2) miembro de la organización que requiere el software y es el responsable de conducir el proyecto desde la perspectiva del negocio. Es quien debe comunicar una visión clara del producto y definir sus características principales, priorizando los requerimientos del cliente para que el proceso de desarrollo se centre en aquellos requerimientos necesarios para la organización. En todo momento el *ProductOwner* debe contribuir con el equipo (*ScrumTeam* en Fig. 2) para remover todas las dudas que surgen acerca de los requerimientos. Por esta razón, es necesario que el lugar de trabajo del *ProductOwner* esté en el mismo espacio físico que el equipo.

El *ScrumMaster* (Fig. 2) es el encargado de garantizar que los principios de Scrum son aplicados en el proceso de desarrollo. Su función es asegurar que el equipo tenga el conocimiento, las habilidades y la cantidad de personas necesarias para llevar a cabo el trabajo requerido. Usualmente desempeña este rol un individuo que es Scrum Master certificado, un experto en Scrum, para asegurarse que los principios de Scrum sean aplicados.

El *DevelopmentTeam* (Fig. 2) es el equipo de trabajo responsable del desarrollo del producto a entregar, para esto trabaja en conjunto con los distintos stakeholders desde la definición de los requerimientos (*ProductBacklog*, definido en Sección 2.3) hasta la entrega del producto. El tamaño del equipo es una cuestión fundamental para los procesos ágiles, siendo lo común equipos entre 3 y 9 desarrolladores por equipo (relación de agregación en Fig. 2 entre *DevelopmentTeam* y *Developer*). Con más de 9 personas la cantidad de relaciones entre los integrantes aumenta exponencialmente y esto es caótico para la comunicación del equipo [11][12]. El *DevelopmentTeam* puede incluir al *ScrumMaster* y al *ProductOwner* solo en los casos que intervengan en la ejecución de trabajo para el *SprintBacklog* [9] (definido en la Sección 2.3).

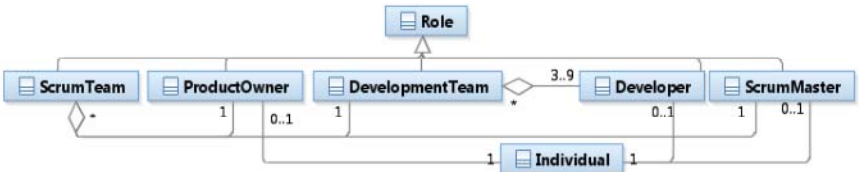

Fig. 2. *Roles* definidos en *Scrum*.

2.2 Eventos

Los eventos (*Event* en Fig. 3) son ocurrencias en el tiempo de un conjunto de reuniones o acciones, utilizados con el objetivo de sincronizar las diferentes etapas por las que atraviesa un proceso Scrum. En La Fig. 3 se representa la relación entre los eventos mediante la asociación *predecesor – successor*. Todo *Event* ocurre en un determinado intervalo de tiempo, representado por *TimeBox* y la asociación *AllocatedOn* en Fig. 3. Un *TimeBox* es una técnica empleada en Scrum para fijar un límite de tiempo en la duración de cada evento o reunión dentro del proceso.

El evento principal en Scrum es un *Sprint*. Un *Sprint* ocurre en un *TimeBox* de 1 mes como máximo y cumple la función de contenedor de eventos. Es utilizado para cumplir los objetivos definidos en él, para generar productos y también para analizar e inspeccionar el proceso. Puede ser analizado como un proyecto debido a que conlleva un esfuerzo temporal para crear un producto y el final se alcanza por el cumplimiento del objetivo o la finalización del proyecto. Un *Sprint* es sucedido por otro, inmediatamente después de que el predecesor es finalizado.

Las distintas reuniones dentro de un *Sprint* se suelen completar de manera secuencial y se estructuran en pequeños *TimeBox*. Esto permite la división del proceso en subconjuntos lógicos que facilitan su dirección, planificación y control. La entidad *Sprint* es llevado a cabo mediante los eventos *SprintPlanning* Meeting, *DailyScrum*, *Sprint Review*, *SprintRetrospective* y la actividad más importante en Scrum que es el desarrollo mismo del producto, *DevelopmentWork* en Fig. 3.

El *SprintPlanning* es una reunión donde se plantea como objetivo elicitar los requerimientos del cliente y la planificación de qué artefactos serán entregados y cómo se construirán. La entidad *SprintPlanning* (Fig. 3) posee un *TimeBox* de ocho horas para un *Sprint* de un mes y tiene una duración proporcionalmente menor para *Sprint* más cortos. Está compuesta por dos partes (*FirstPart* y *SecondPart* en Fig. 3), cada una con la mitad de la duración de toda la reunión *SprintPlanning*. En la reunión *FirstPart* (Fig. 3) se desarrolla el qué se hará durante el *Spring*. El objetivo de esta reunión es que el *DevelopmentTeam* entienda en detalle los requerimientos del usuario. Con esto, para finalizar esta etapa deciden cuales requerimientos están en condiciones para ser desarrollados. En la reunión *SecondPart* (Fig. 3) se desarrolla el "cómo" se obtendrán los mismos. El objetivo de esta actividad es que el *DevelopmentTeam* determine qué interfaces necesitará implementar, qué arquitectura deberá crear y que tablas o componentes requerirán ser actualizados o desarrollados.

El *DailyScrum* es una reunión corta donde el *DevelopmentTeam* coordina y planea su siguiente día de trabajo reportando avances y dificultades. Un *DailyScrum* (Fig. 3)

ocurre en un *TimeBox* de 15 minutos realizado cada día de trabajo. Este evento es seguido por el *DevelopmentWork***,** la parte más importante del *Sprint* donde se implementa la solución. Además, en esta etapa se realizan las pruebas y al finalizar se entrega el incremento logrado en el producto (*ProductIncrement*, descripto en Sección 2.3). Luego del *DevelopmentWork* sucede una *SprintReview*. Esta es una reunión en la que se inspecciona el incremento de producto (*ProductIncrement*) creado y se adapta el *ProductBacklog* (explicados en Sección 2.3) para el siguiente *Sprint*. Como resultado del *SprintReview* se obtiene información de entrada para el siguiente *SprintPlanning*. Generalmente un *SprintReview* sucede en un *TimeBox* de cuatro horas para una *Sprint* de un mes.

La otra reunión de inspección posible en un *Spring* es la *SprintRetrospective*, la cual se desarrolla luego de la *SprintReview* y antes de la próxima reunión de *SprintPlanning* (ver Fig. 3). Este evento tiene como objetivo revisar cómo fue realizado el último *Sprint* (*Roles*, *Eventos*, y *Artefactos*). En ella se crea un plan de mejoras que serán aplicadas durante la siguiente iteración. Usualmente sucede en un *TimeBox* de tres horas para una *Sprint* de un mes.

Durante un Sprint puede surgir un evento denominado *Grooming* (Fig. 3). El *Grooming* es una especificación (división) de una tarea en tareas más pequeñas.

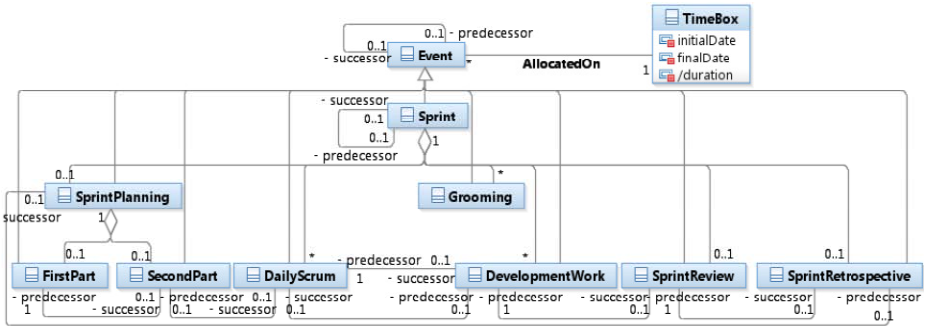

Fig. 3. Representación de eventos definidos en Scrum.

2.3 Artefactos

Un artefacto es la descripción de un producto de trabajo. Los artefactos pueden estar compuestos por otros artefactos y son productos de trabajos concretos consumidos, producidos o modificados por los distintos eventos del proceso [12]. En la Fig. 4 se representan los conceptos vinculados a los artefactos del proceso.

La generación del producto principal está guiada por un conjunto ordenado de requerimientos denominado *ProductBacklog*. El *product backlog* contiene cada requerimiento (*ProductBacklogItem*) que podría ser tratado en el desarrollo de un producto (*Product*), como así también en cada incremento del producto (*ProductIncrement*). Habitualmente la lista de *ProductBacklogItem* se encuentra ordenada por valor, riesgo, prioridad y necesidad. Asimismo, el incremento del producto sirve luego como retroalimentación, permitiendo la evaluación del

incremento y la generación de nuevos requerimientos. De esta manera se representa un *Feedback* entre *ProductIncrement* y *ProductBacklogItem*. Los *ProductBacklogItem* son requerimientos que plantean necesidades, deseos o expectativas que el *ScrumTeam* quiere entregar en el futuro. Estos ítems son parte del *ProductBacklog* y a su vez pueden ser refinados (*RefinedOn*) en nuevos elementos durante un evento *Grooming* (Fig. 3).

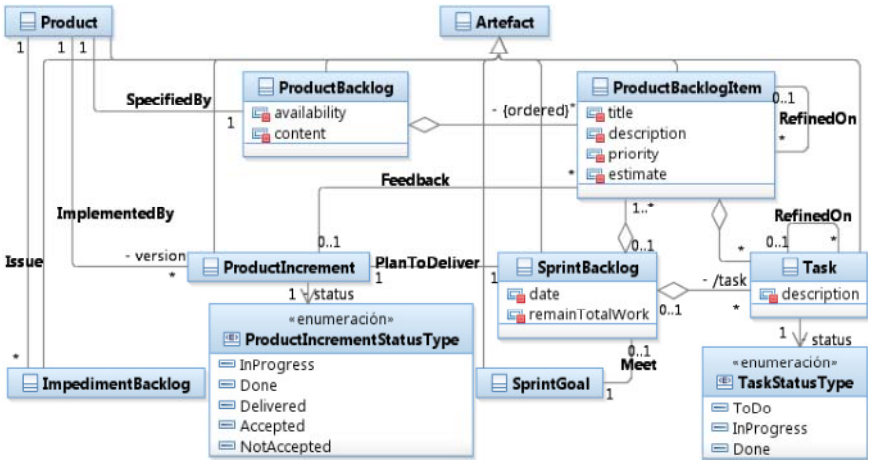

Fig. 4. Representación de Artefactos de Scrum.

El *SprintBacklog* es una lista de tareas (*Task*) que indica las actividades que se tienen que realizar durante el *Sprint* y cuál es la situación actual del equipo de desarrollo. El *SprintBacklog* reúne al conjunto de *ProductBacklogItem* seleccionados para ser trabajados durante un *Sprint* y con ello poder cumplir el *SprintGoal*, además comprende su planificación para entregar el *ProductIncrement* (relación *PlanToDeli*ver).

El *ProductIncrement* es el resultado de la implementación de todos los *ProductBacklogItem* especificados por el *SprintBacklog*, durante un *Sprint.* Al finalizar el Sprint el *ScrumTeam* entrega esta versión parcial del producto de software. Este entregable tiene una propiedad *status* la cual representa el estado en que se encuentra (*ProductIncrementStatusType*): *InProgress*, *Done*, *Delivered*, *Accepted*, o *NotAccepted*.

Los distintos ítems del *ProductBacklog* que componen el *SprintBacklog* se descomponen en un conjunto de tareas (*Task* en Fig. 4) que indican lo que el equipo de desarrollo debe realmente hacer para lograr el *SprintGoal*. Durante el proceso de desarrollo cada tarea puede estar en el *status* (*TaskStatusType* en Fig. 4): *ToDo*, *InProgress*, o *Done*. *ToDo* representa una tarea del *SprintBacklog* que está sin realizarse. Si permanece más de un día de trabajo en este estado es fragmentada en tareas más pequeñas [13] (relación *RefinedOn*). *InProgress* es cuando la tarea fue comenzada pero aun no finalizada. *Done* representa que la tarea fue finalizada.

Las distintas dificultades presentes en el proceso se incorporan en el *ImpedimentBacklog* , una lista de dificultades, impedimentos u obstáculos que limitan el rendimiento del *DevelopmentTeam*. El *ScrumMaster* tiene la responsabilidad de removerlos lo antes posible.

2.4 ¿Qué Eventos generaron un Artefacto en particular?

Los eventos son secuenciales y la conclusión de un evento finaliza con la entrega de un artefacto producido (relación *Output* en Fig. 1). Así también la finalización de un evento representa un punto a ser evaluado y adaptado. Cada artefacto (Fig. 4) es una herramienta o un producto de trabajo intermedio que permite llevar a cabo el trabajo en entornos difíciles. Cada *Sprint* tiene una visión única que la hace diferente de cualquier otra que pudiera sucederla e intenta alcanzar un cierto objetivo (*SprintGoal* en Fig. 5). En la Fig. 5 se representan las entradas y salidas de un *Sprint*. El modelo es extensible a los distinots eventos que componen el *Sprint*. En Fig. 5 se incluye a modo de ejemplo los eventos *FirstPart* del *SprintPlanning* y *DailyScrum*.

Durante la reunión *FirsPart* del *SprintPlanning* se define el objetivo del *Sprint*, *SprintGoal*, y los *ProductBacklogItem* que serán desarrollados durante la *Sprint* en curso, representados por la relación *Output* de *FirstPart* en Fig. 5.

En la realización del *DailyScrum* se coordina y planea el siguiente día de trabajo reportando avances y dificultades, se seleccionan los elementos en los que se trabajará, representado por la relación *Satisfy ProductBacklogItem* (Fig. 5), y se intenta resolver las dificultades presentes (relación *Remove ImpedimentBacklog* en Fig. 5). Las relaciones *Satisfy* y *Remove* especializan la relación *Output* definida entre *Event* y *Artefact* en Fig. 1.

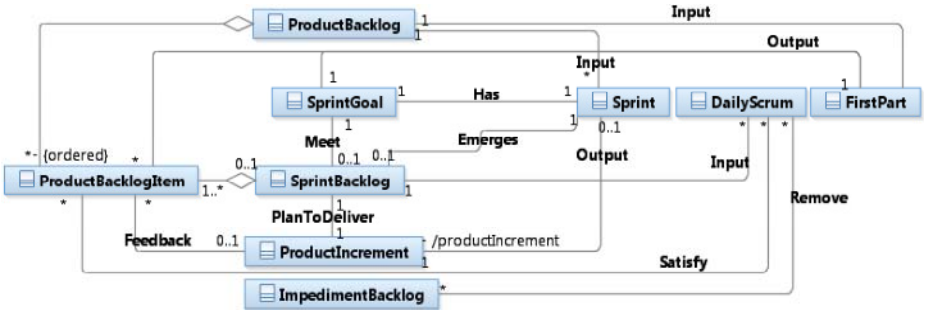

Fig. 5. Representación de Artefactos vinculados a los Eventos *Sprint*, *FirstPart*, y *DailyScrum*.

2.5 ¿Qué Requerimientos Guiaron la Generación de cierto Artefacto?

Los eventos representados en el modelo propuesto junto a las relaciones con los distintos artefactos de un proceso Scrum permiten obtener información acerca de qué requerimientos fueron considerados en la generación de los distintos artefactos, principalmente de los incrementos del producto principal de software (*ProductIncrement* en Fig. 5). En Scrum cada *Sprint* posee un objetivo (*SprintGoal*)

el cual es materializado por un *SprintBacklog*, el cual reúne un conjunto de requerimientos detallados en el *ProductBacklogItem*. La representación explícita de estas relaciones (*Has*, *Meet*, y *Emerges* en Fig. 5) permite conocer los requerimientos que guiaron los distintos *Sprint* del proceso de desarrollo, en los cuales se generaron los distintos incrementos del producto (*ProductIncrement* en Fig. 5).

2.6 ¿Quiénes son los Participantes Involucrados en un Evento dado?

Como fue mencionado previamente, los eventos son realizados (*Perform* en Fig. 1) por uno o más individuos (*Individual* en Fig. 2) ejerciendo ciertos *Roles*. La relación *Perform* explicitada en la Fig. 1 es especializada para representar las distintas formas de participación en el proceso. Por limitaciones de espacio en esta sección sólo se presenta la especialización para el evento *Sprint*. La Fig. 6 incluye dicho evento y los distintos roles que participan del mismo.

Una de las principales responsabilidades del *ProductOwner* es definir qué se debe realizar en el *Sprint* (relación *Define* en Fig. 6). Selecciona los cinco *ProductBacklogItems* más importantes, son asignados al *Sprint*, y se asegura que el equipo pueda desarrollarlas en el *Sprint*. La otra responsabilidad que posee es revisar y aprobar el trabajo en el *Sprint* diciendo si lo que fue creado o "hecho" satisface el objetivo del *Sprint* (relación *Aprove* en Fig. 6).

La función primordial del *ScrumMaster* es asegurar que las políticas de la organización sean adoptadas por el *DevelopmentTeam* y de revisar cómo fue realizado el último Sprint, los *Roles*, *Eventos*, y *Artefactos* que lo componen.

El *ScrumMaster* debe trabajar junto al *ProductOwner* y ambos tienen la responsabilidad de intermediar para lograr un entendimiento o negociación sobre algún problema en los tiempos de trabajo o agregar *ProductBacklogItem* al *Sprint*.

El *DevelopmentTeam* participa dentro de un *Spring* realizando todo el *DevelopmentWork*, toma los requerimientos, los completa y satisface, creando un producto de software. Realiza las tareas de análisis, diseño, desarrollo, testing, documentación y mantenimiento. El equipo decide que será realizado y por cuál actor dentro del equipo. Se auto organiza para planear cómo va a realizar el trabajo dentro del *Sprint*, esto es un punto crítico de Scrum.

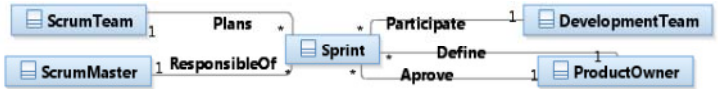

Fig. 6. Representación de los *Roles* participando en un *Sprint*.

3 Caso de Estudio

El modelo presentado en la Sección 2 se lo aplica a un proyecto de sistema de comercio electrónico. El *ProductOwner* presenta los *ProductBacklogItem* que considera que deben ser estimados. El *ScrumTeam* divide los ítems del *Backlog* en ítems más pequeños (*Grooming*), y redimensiona el *ProductBacklog* con los nuevos

elementos; por ejemplo, en la Fig. 7 se ilustra el *ProductBacklog*, donde el ítem (*ProductBacklogItem*) *Checkout* fue refinado (*RefinedOn*) en los ítems *Selección Formas de Pago* y *Selección Medio Envío.*

En el evento *FirstPart* del *Sprint* 1 el *ProductOwner* ordena el *ProductBacklog* y define cuál será el *SprintGoal,* en este caso *Diseño Frontend*, *Administrar Productos*, y *Diseño Base de Datos.* En la *SecondPart,* el *DevelopmentTeam* realiza bosquejos para crear o actualizar las interfaces, la arquitectura y/o los componentes. Además, define los *ProductBacklogItem* a desarrollar en el *Sprint1* (*SprintBacklog* en Fig.8).

La Fig. 8 presenta el *SprintBacklog* junto a información de los *ProductBacklogItems* y *Tasks* que lo componen. La figura muestra el backlog una vez iniciada la tarea de "Diseñar Header, Body, Footer" por el *DevelopmentTeam*, donde el *Status* de la tarea fue pasado de *ToDo* a *InProgress*.

	Title	ID	Owner	Priority
	Sprint1			
	Diseño Frontend	S-01010		High
	Administrar Productos	S-01004		High
	Diseño Base de Datos	S-01012		Medium
	Sprint2			
Checkout	Seleccion Formas de Pago	S-01003		High
	Seleccion Medio Envío	S-01013		High
	Implementación del carro	S-01008		High
	Administrar Cuenta de Usuario	S-01002		Low

Title	ID	Owner	Priority
Sprint3			
Administrar Pedidos	S-01007		Medium
Ingreso al Sistema	S-01001		Low
Administrar Usuarios	S-01005		Low
Diseño Backend	S-01011		Low

Fig. 7. *ProductBacklog* del caso de estudio.

PBI	RemainTotalWork	Title	PBI-ID	TaskID	Estimate	Priority	Status
Diseño Frontend	50,00%	Diseñar Header, Body, Footer	S-01010	T-0001		High	InProgress
		Crear el Estilo	S-01010	T-0002			Done
Administrar Productos	100,00%	Crear Productos	S-01004	T-0003		High	InProgress
		Actualizarr Productos	S-01004	T-0004			ToDo
		Eliminar Productos	S-01004	T-0005			ToDo
Diseño Base de Datos	100,00%	Crear Tablas y Atributos	S-01012	T-0006		Medium	InProgress

Fig. 8. *SprintBacklog* de la iteración número 1.

Una vez concluido el *Sprint 1,* el *DevelopmentTeam* entrega los *ProductBacklogItem* con *status Done* al *ProductOwner.* Este último es el encargado de inspeccionar si cumple con la definición de producto terminado, aprobando o rechazándolo. Una vez aprobado el *ProductBacklogItem*, el *ScrumMaster* organiza el *Sprint Review*. Además toma nota del *Feedback* del encuentro del *ProductOwner* con el usuario final, donde es presentado el *ProductIncrement.* Si este es aprobado el *ScrumTeam* libera *e-Commerce versión 0.1*.

Durante el *SprintRetrospective* el *ScrumTeam* evalúa el *Sprint1,* registrando los logros alcanzados, las dificultades que surgieron y las sugerencias de cómo mejorar el proceso. Una vez finalizado esto, el *ScrumTeam* se da inicio al *Sprint2*.

Para poder validar la aplicación del modelo propuesto al ejemplo presentado se utilizó la herramienta USE (UML Specification Environment) la cual permite validar y verificar especificaciones consistentes de diagramas de clases UML y realizar consultas en OCL [14]. La validación se realizó generando las instancias del modelo propuesto siguiendo la ejecución previamente enunciada del caso de estudio. A partir

del mismo, fue posible conocer los requerimientos (instancias de *ProductBacklogItem*) considerados en el incremento de producto *e-Commerce versión 0.1*, como así también los miembros del *Scrum Team* que participaron en el *Sprint 1* que generó tal artefacto.

4 Conclusiones

En este trabajo se propone un modelo conceptual de Scrum para representar los eventos que lo componen junto a los artefactos generados. En dicho modelo se establecieron diferentes conceptos que especializan los conceptos básicos de Scrum y que pueden ser utilizados para asistir en tareas de trazabilidad en proyectos ágiles tales como: Stakeholders con Requerimientos, User Stories con Versiones y Requerimientos con Requerimientos. Los trabajos futuros utilizarán el modelo conceptual propuesto para formalizar el soporte a tales tareas.

5 Referencias

1. IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990 1–84 (1990)
2. Espinoza, A., Garbajosa, J.: A study to support agile methods more effectively through traceability. Innovations in Systems and Software Engineering 7, 53–69 (2011)
3. Pikkarainen, M., Passoja, U.: An Approach for Assessing Suitability of Agile Solutions: A Case Study. In: 6th International Conference on Extreme Programming and Agile Processes in Software Engineering (2005)
4. Agile Manifesto, http://www.agilemanifesto.org (2001)
5. Williams, L.: What agile teams think of agile principles. ACM Communications 55, 71–76 (2012)
6. Hunt, J.: Agile Software Construction. Springer (2005)
7. Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.: A decade of agile methodologies: Towards explaining agile software development. Journal of Systems and Software 85, 1213–1221 (2012)
8. Cao, L., Ramesh, B.: Agile Requirements Engineering Practices: An Empirical Study. IEEE Software 25, 60–67 (2008)
9. Sutherland, J., Schwaber, K.: The Scrum Guide. The Definitive Guide to Scrum: The Rules of the Game (2011)
10. Schwaber, K.: SCRUM Development Process. In: OOPSLA'95 Workshop on Business Object Design and Implementation (1995)
11. Brooks , F.: Mythical Man-Month, The: Essays on Software Engineering, 2nd ed. Addison-Wesley Professional (1996)
12. OMG: Software & Systems Process Engineering Metamodel Specification (SPEM) Version 2.0 (2008)
13. Glogler, B.: Scrum Checklist 2012. bor!sgloger Wien. Baden-Baden. (2012)
14. P. Ziemann, M. Gogolla: Validating OCL Specifications with the USE Tool – An Example Based on the BART Case Study. Electronic Notes in Theoretical Computer Science 80, 157-169 (2003)