Un Modelo Conceptual para la Trazabilidad de Procesos Scrum

Nazareno, Roberto[1, 2]; Leone, Horacio[1, 3]; Gonnet, Silvio[1,3]
[1]INGAR (CONICET – UTN)
[2]Universidad Nacional de La Rioja
[3]Universidad Tecnológica Nacional, Facultad Regional Santa Fe

Abstract
En este trabajo se formula un modelo conceptual del proceso Scrum con el objetivo de asistir en las tareas de trazabilidad en proyectos ágiles. El modelo permite explicitar las trazas de Stakeholders con Requerimientos, User Stories con Versiones y Requerimientos con Requerimientos. Así mismo en dicho modelo se representan diferentes perspectivas del Framework, brindando soporte a prácticas básicas de trazabilidad que pueden ser interpretadas como "trazabilidad en procesos agiles". Este trabajo propone responder a preguntas de competencia tales como: 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 estas preguntas empleando el modelo propuesto facilitarían prácticas de trazabilidad sin perder agilidad.

Palabras Clave
Scrum, Trazabilidad, Requerimiento.

1. Introducción

La trazabilidad es considerada en metodologías ágiles como un aspecto fundamental a estudiar para desarrollar sistemas de calidad [1] [2]. Trazabilidad es: (i) el grado en el cual se puede establecer una relación entre dos o más productos del proceso de desarrollo (traza), 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 [3]. Las metodologías de desarrollo "pesadas" centraron sus prácticas de trazabilidad en el establecimiento de trazas desde requerimientos a otros artefactos de desarrollo. Por otro lado, 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 el desarrollo de especificaciones no ambiguas y completas es imposible o incluso no apropiado [1], 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 tests 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: ¿Quiénes son los stakeholders responsables de los Artefactos?, ¿De cuáles Artefactos es responsable cada Stakeholder?, ¿En qué Evento es definido cada Artefacto?, ¿Qué Artefactos se crean durante cada Evento?, ¿de qué Evento es responsable cada Stakeholder?, ¿en qué Eventos participa cada Stakeholder?, Si existe un cambio en un requerimiento, ¿sobre qué actores o en

qué eventos repercute?, Si existe un problema en el desarrollo, ¿qué eventos, Artefactos y Stakeholder estarían involucrados? Las respuestas a estar preguntas asistirían las tareas de traza en proyectos ágiles tales como: Stakeholders con Requerimientos, Requerimientos 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 [1].

La siguiente sección presenta el modelo de Scrum propuesto. El modelo es estructurado en 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, discutiendo el alcance de la propuesta. Por último, en la Sección 4, se presentan las conclusiones del trabajo.

2. Modelo para la Trazabilidad de Procesos 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. Para lograr vistas simplificadas del modelo, estas reglas se presentan combinando los conceptos de a pares: *roles con artefactos*, *artefactos con eventos*, y *eventos con roles*; obteniendo de esta manera, diferentes perspectivas del modelo.

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 define los roles: *ScrumTeam*, *ProductOwner*, *ScrumMaster*, y *Developmeant Team* (Figura 1). Para mantener la trazabilidad de un proceso ágil es preciso modelar los roles debido a que son estos los que tendrán la responsabilidad de llevar a cabo diferentes actividades durante la ejecución del proceso.

Scrum Team. Es un equipo de trabajo integrado por diferentes participantes del proceso de desarrollo: el *ProductOwner*, el *ScrumMaster* y el *DevelopmentTeam* (Figura 1). Los miembros del equipo son individuos de la organización desarrolladora del software y de la organización que requiere el software.

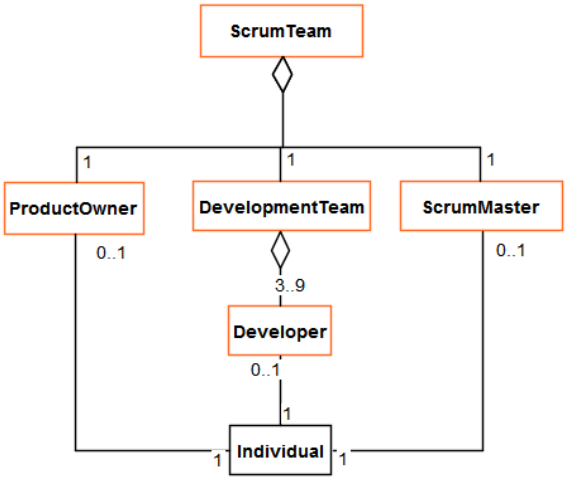

Figura 1. *Roles* definidos en *Scrum*.

Product Owner. El *ProductOwner* (Figura 1) es un individuo (*Individual* en Figura 1) parte 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. Es decir, su función principal es priorizar los requerimientos del cliente para que el proceso de desarrollo se centre en aquellos ítems del *Backlog* (definido en Sección 2.3) necesarios para la organización. En todo momento el *ProductOwner* debe contribuir con el equipo (*ScrumTeam* en Figura 1) 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.

Scrum Master. Es un individuo (*Individual* en Figura 1) 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.

Development Team. El *DevelopmentTeam* (Figura 1) es el equipo de trabajo responsable 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 Figura 1 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).

2.2. Eventos

Los eventos (*Event* en Figura 2) 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. Es importante en el modelado debido a que son en estos en donde la trazabilidad tiene lugar en el espacio temporal. En la Figura 2 se representa la relación entre los eventos mediante la asociación *predecesor – successor*. Toda reunión ocurre en un determinado intervalo de tiempo, representado por *TimeBox* y la asociación *AllocatedOn* en Figura 2.

Time Box. Un *TimeBox* (Figura 2) es una técnica empleada en Scrum para restringir la duración de cada evento o reunión dentro del proceso Scrum. Se define la propiedad *duration* como una propiedad derivada de *initialDate* y *finalDate* con la duración en *dateTime* del TimeBox.

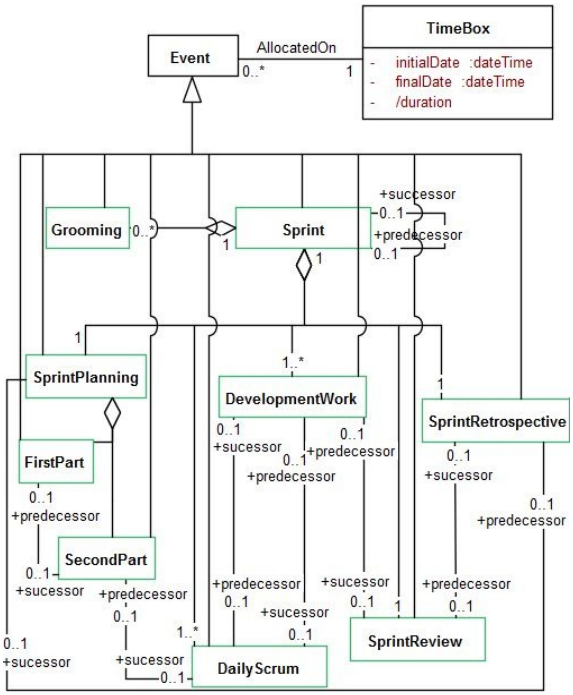

Figura 2. Representación de tipos de eventos definidos en Scrum.

Sprint. Un *Sprint* es el *Event* principal de Scrum y cumple con 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 (relación *predecessor - successor* en Figura 2). Usualmente un *Sprint* posee un *TimeBox* con una duración de 1 mes como máximo.

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*, *DailyScrum*, *SprintReview*, *SprintRetrospective* y la actividad más importante en Scrum que es el desarrollo mismo del producto, *DevelopmentWork* en Figura 2.

Sprint Planning. 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* (Figura 2) es un *Event* que ocurre en 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 Figura 2), cada una con la mitad de la duración de toda la reunión *SprintPlanning*.

First Part. En esta reunión se define qué se hará durante el *Sprint*. El objetivo de esta reunión es que el *DevelopmentTeam* entienda en detalle los requerimientos del usuario. Con esto, para finalizar esta etapa deciden cuáles requerimientos están en condiciones para ser desarrollados.

Second Part. En esta parte se desarrolla el cómo se hará el *Sprint*. El objetivo de esta etapa es que el *DevelopmentTeam* diseñe que interfaces necesitará implementar, qué arquitectura deberá crear y que tablas o componentes requerirán ser actualizados o desarrollados.

Daily Scrum. Es una reunión corta donde el *DevelopmentTeam* coordina y planea su siguiente día de trabajo reportando avances y dificultades. Un *DailyScrum* (Figura 2) es un *Event* que ocurre en un *TimeBox* de 15 minutos realizado cada día de trabajo, a la misma hora y en el mismo lugar.

Development Work. Es la parte del *Sprint* dedicada a la implementación de 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).

Sprint Review. 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*. Se realiza una vez finalizado el desarrollo (el *DevelopmentWork* en Figura 2). Como resultado del *SprintReview* se obtiene información de entrada para la siguiente reunión de *SprintPlanning*. Un *SprintReview* es un *Event* que usualmente ocurre en un *TimeBox* de cuatro horas para una *Sprint* de un mes y tiene una duración proporcionalmente menor para *Sprint* más cortos.

Sprint Retrospective. Es una reunión de inspección que 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 (el siguiente *Sprint*). Es un *Event* que ocurre en un *TimeBox* usualmente de tres horas para una *Sprint* de un mes y tiene una duración proporcionalmente menor para *Sprint* más cortos. La *SprintRetrospective* se desarrolla luego de la *SprintReview* y antes de la próxima reunión de *SprintPlanning*.

Grooming. El *Grooming* es una actividad llevada a cabo durante un *Sprint* (ver Figura 2) para especificar o refinar una tarea (*Task* explicado en sección 2.3) o ítem (*ProductBacklogItem* explicado en sección 2.3) en tareas o ítems más pequeños. De esta forma el *DevelopmentTeam* o el *ProductOwner* pueden añadir detalles, estimaciones y ordenar los ítems por prioridad. Normalmente este evento no consume más del 10% del tiempo total asociado al *Sprint*.

2.3. Artefactos

Un artefacto es la definición de un producto de trabajo. Los artefactos pueden estar compuestos por otros artefactos y son productos de trabajo concretos consumidos, producidos o modificados por los distintos eventos del proceso [12]. En la Figura 3 se

representan los conceptos vinculados a los artefactos de un proceso Scrum.

Product Backlog. Es una lista ordenada que contiene todos los requerimientos (*ProductBacklogItem* en Figura 3) que podrían ser tratados para el desarrollo de un producto, en particular de cada incremento del producto (*ProductIncrement* en Figura 3). En consecuencia, la entidad *ProductBacklog* está relacionada mediante una relación todo-parte con la entidad *ProductBacklogItem*. Habitualmente la lista de producto se encuentra ordenada por valor, riesgo, prioridad y necesidad.

Product Backlog Item. Los *ProductBacklogItem* son requerimientos que plantean necesidades, deseos o expectativas que el *ScrumTeam* quiere entregar en el futuro. Estos *Items* son parte del *ProductBacklog* y a su vez pueden ser refinados (asociación *RefinedOn* en Figura 3) por el *DevelopmentTeam* o el *ProductOwner* en nuevos elementos durante un evento *Grooming*. Los *ProductBacklogItem* tienen como propiedades el nombre (*title*), la descripción del requerimiento (*description)*, la prioridad (*priority*) y la estimación del requerimiento (*estimate*).

Sprint Backlog. Un *SprintBacklog* es una lista de tareas (*Task*) que indica las actividades que el equipo de desarrollo tiene que realizar durante el *Sprint*. El *SprintBacklog* reúne al conjunto de *ProductBacklogItem* (relación “todo-parte” en Figura 3) seleccionados para ser trabajados durante un *Sprint* y con ello poder cumplir el *SprintGoal*. Además, comprende su planificación para entregar el *ProductIncrement* (asociación *PlanToDeliver* en Figura 3). El *SprintBacklog* no refleja el tiempo invertido en cada tarea sino que representa el total de tareas por realizar manteniendo la situación actual del equipo de desarrollo.

Product Increment. 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. 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*. El entregable tiene una propiedad *status* la cual representa el estado en el que se encuentra (*ProductIncrementStatusType*):

- *InProgress*: es cuando el *DevelopmentTeam* comienza a crear el incremento al inicio del *Sprint*.
- *Done*: es cuando el incremento cumple con la definición de finalizado creada por el *ScrumTeam.* Se encuentra en este estado hasta que el *ProductOwner* lo presente para ser evaluado durante el *SprintReview*.
- *Delivered*: es un estado intermedio o de transición desde el incremento con *status Done* es entregado para ser inspeccionado, permitiendo la retroalimentación de información, pudiendo incluir nuevos *ProductBacklogItem* (asocación *Feedback* en Figura 3).
- *Accepted*: es presentado el incremento y es aceptado para ser liberado como una nueva *versión* de *Product*. Esta versión es aditiva a todos los Incrementos anteriores con una funcionalidad potencialmente utilizable.
- *NotAccepted*: el incremento es presentado y no aceptado por el *ProductOwner* para ser liberado como una nueva *versión* de *Product*. Esto puede suceder debido a que el *SprintGoal* hubiese quedado obsoleto o que cambiasen las reglas de negocio de la compañía durante el desarrollo del *Sprint*.

Task. Una *tarea* es la unidad mínima de trabajo que compone a un *ProductBacklogItem* (relación “todo-parte” en Figura 3) e indican lo que el *DevelopmentTeam* debe realmente hacer para lograr el cumplimiento del *SprintGoal*. Durante el proceso de desarrollo cada tarea puede estar en el *status* (*TaskStatusType*):

- *ToDo*: es cuando una tarea del *SprintBacklog* está sin realizar. Si

permanece más de un día de trabajo en este estado es fragmentada en tareas más pequeñas [13].

— *InProgress*: es cuando un desarrollador comenzó a realizar la tarea. Una vez comenzada, pero aun no finalizada, el *ScrumMaster* posee la responsabilidad de marcarla en este estado y de anotarla en su lista de dificultades. Si permanece más de un día de trabajo en este estado es fragmentada en tareas más pequeñas [13] (relación *RefinedOn* en Figura 3).

— *Done*: la tarea se marca como completa cuando cumple con la definición de tarea finalizada, creada por el *DevelopmentTeam* en la *SecondPart* del *SprintPlanning*. En el porcentaje de progreso no existe crédito o un progreso parcial, está completa o no.

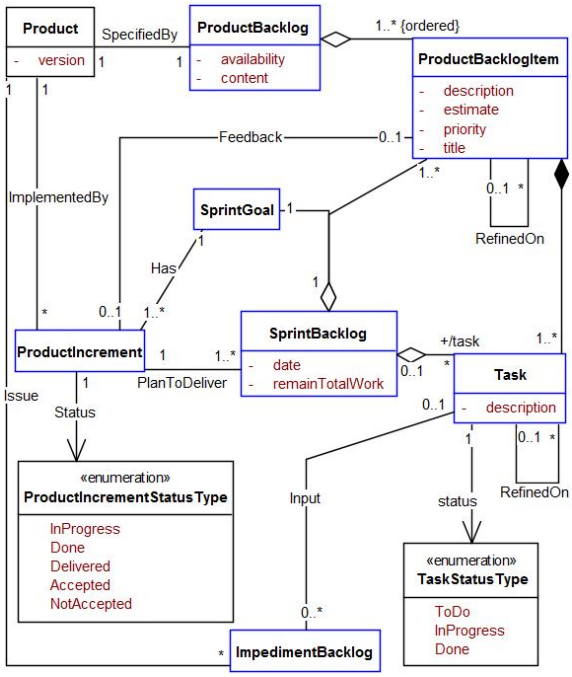

Figura 3. Representación de Artefactos de Scrum.

Impediment Backlog. El *ImpedimentBacklog* es una lista de dificultades, impedimentos u obstáculos que limitan el rendimiento del *DevelopmentTeam*. Si una *Task* fue comenzada y no fue finalizada durante el *DevelopmentWork* es registrada como un impedimento (relación *Input* en Figura 3). El *ScrumMaster* tiene la responsabilidad de darles una solución lo antes posible.

Sprint Goal: es el resultado deseado que deberá satisfacer el *DevelopmentTeam* con la implementación de los requerimientos y funcionalidades necesarias en un *ProductIncrement* durante el presente *Sprint*.

2.4. Relaciones entre Roles y Artefactos

En Scrum los *artefactos* son responsabilidad de un único *rol*, esto simplifica la trazabilidad de los responsables de los distintos artefactos. Sin embargo, los otros *roles* pueden emplear los *artefactos*, y tal vez hasta actualizarlos, si es que el rol tiene la autorización de hacerlo. En la Figura 4 se representan las relaciones entre los *Roles* identificados en Scrum y los *Artefactos* del proceso de desarrollo.

Uno o varios *ScrumTeam* trabajan en una organización para el desarrollo de un *Product* (relación *Release* en Figura 4). Cada *Product* siempre tendrá disponible una *version* en funcionamiento.

La principal tarea o rol del *ProductOwner* es la de desarrollar la visión del producto y plasmarla en el *ProductBacklog*, esto es entender el qué quiere el cliente. Él no desarrolla la visión solo, sino junto a los distintos participantes del cliente. En consecuencia, el *ProductOwner* es el responsable de administrar el *ProductBacklog* (asociación *ResponsibleOf* en Figura 4). Él crea las “user stories” (*ProductBacklogItem* en Figura 4) y prioriza (ordena) los *Items* del *Backlog* (relación *Create* Figura 4), para alcanzar los objetivos del Sprint (*SprintGoal*).

El principal objetivo de un *ScrumMaster* es remover obstáculos o impedimentos del *ImpedimentBacklog* (asociación *Resolve* en Figura 4), se encarga de facilitar herramientas al *DevelopmentTeam*, ayudándolo cuando tenga algún impedimento. Por consiguiente debe escucharlos para poder remover los obstáculos que les impiden llevar a cabo su tarea. Asimismo debe asegurarse que el *DevelopmentTeam* tenga el conocimiento, habilidades y la cantidad de personas

necesarias para lograr el trabajo. El *ScrumMaster* se encarga de administrar los *ProductBacklogItem* (relación *Manage* en Figura 4).

El *DevelopmentTeam* es el rol que más interacciona con los artefactos o que hace uso de ellos. Este rol tiene la responsabilidad de organizar su trabajo para satisfacer el *SprintGoal* (asociación *Satisfy* en Figura 4) esto lo logra creando el incremento del producto (asociación *Create* en Figura 4) de acuerdo a lo que se comprometió a entregar en el *Sprint* (relación *ResponsibleOf* en Figura 4). Para poder cumplir con esto, debe desarrollar los *ProductBacklogItem* (asociación *Develop* entre *DevelopmentTeam* y *ProductBacklogItem* en Figura 4) y las *Task* (relación *Develop* entre *DevelopmentTeam* y *Task* en Figura 4). A su vez debe gestionar los obstáculos o dificultades que limitan su desempeño e impiden llevar a cabo su tarea (relación *Manage* entre *DevelopmentTeam* y *ImpedimentBacklog* en Figura 4).

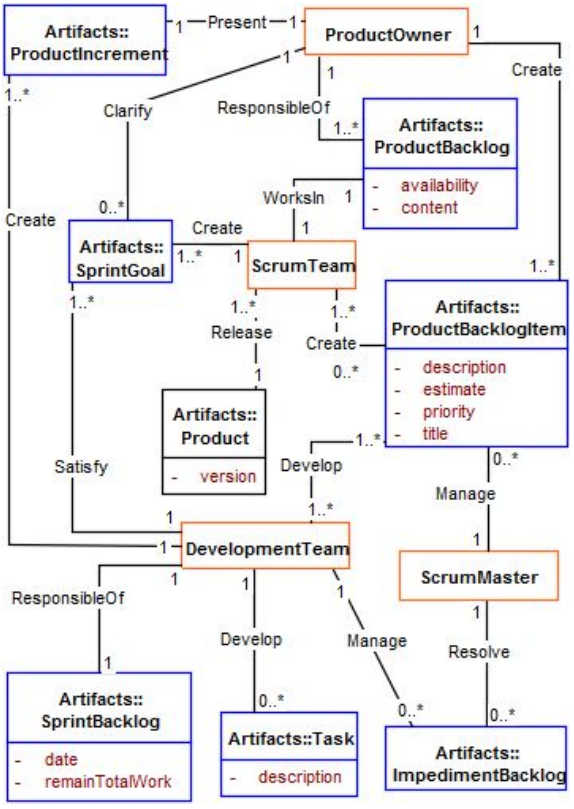

Figura 4. Representación de los Roles y Artefactos.

2.5. Relaciones entre Eventos y Artefactos

Los eventos poseen la característica de que son secuenciales, la conclusión de un evento finaliza con la entrega de un artefacto producido. Así también la finalización un evento representa un punto a ser evaluado y adaptado. Cada artefacto es una herramienta o un producto de trabajo intermedio que permite llevar a cabo el trabajo en entornos difíciles. En la Figura 5 se representan las entradas y salidas de un *Sprint*. Además, se incluyen los distintos eventos que componen a un *Sprint*.

Cada *Sprint* tiene una visión única que la hace diferente de cualquier otra que pudiera sucederla e intenta alcanzar un cierto objetivo (*SprintGoal* en Figura 5).

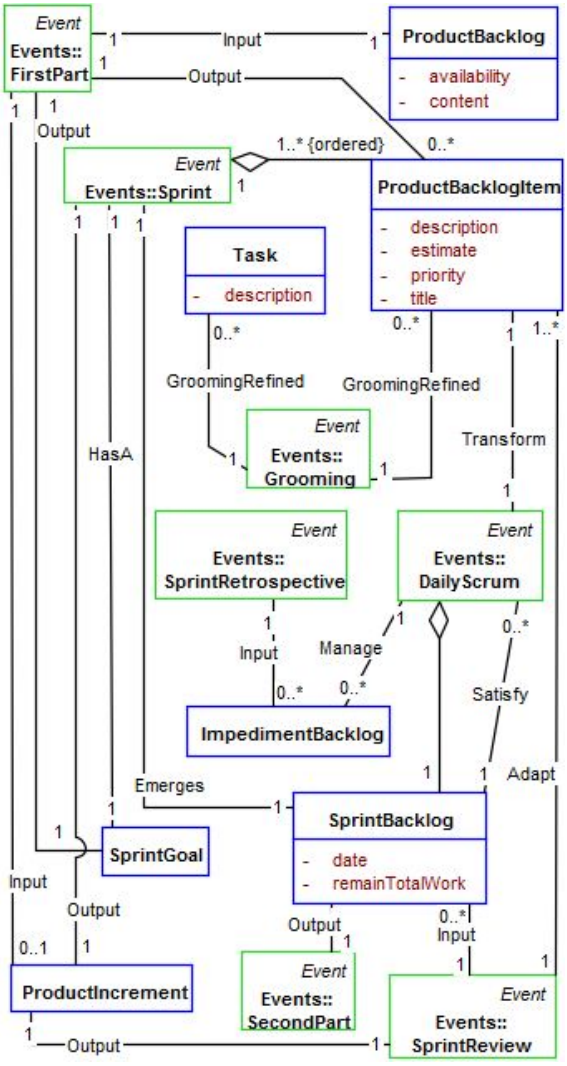

Figura 5. Representación de los Artefactos y Eventos.

El *ProductIncrement* es el resultado de la suma de los *ProductBacklogItem* con s*tatus "done"* creados en el *Sprint* (relación *Output* en Figura 5), y durante todos los *Sprint* anteriores.
Durante la reunión *FirstPart* del *SprintPlanning* se define el objetivo del *Sprint*, *SprintGoal* (asociación *Output* en Figura 5), y los *ProductBacklogItem* que serán desarrollados durante la *Sprint* en curso, representados por la asociación *Output* de *FirstPart* en Figura 5.
Como resultado de la *SecondPart* del *SprintPlanning*, el *SprintBacklog* es creado (relación *Output* de *SecondPart* en Figura 5).
Durante el *DailyScrum* se informa el avance del *ProductBacklogItem* desarrollado el día anterior y lo que se desarrollará hasta el próximo *DailyScrum* (relación *Transform* en Figura 5). De esta manera se puede corroborar si se está cumpliendo con el *SprintBacklog* en curso (asociación *Satisfy* en Figura 5). También aquí se pueden reportar o resolver impedimentos u obstáculos que limitan la tarea diaria de desarrollo (relación *Manage* en Figura 5).
Se da inicio al *SprintReview* una vez que el *remainTotalWork* del *SprintBacklog* es cero (relación *Input* en Figura 5). El punto central de esta reunión es discutir acerca del *ProductIncrement* generado (*status Delivered*). Obteniendo como una salida el *ProductIncrement* con *status Accepted* o *NotAccepted* (relación *Output* en Figura 5). Esta reunión permite una retroalimentación del *ProductIncrement*, pudiendo utilizarse este para actualizar los *ProductBacklogItem* (asociación *Adapt* en Figura 5).
Al final de cada *Sprint* se lleva a cabo el *SprintRetrospective*, esta reunión tiene por objetivo analizar, inspeccionar, discutir lo sucedido durante todo el *Sprint* (relación *Inspect* en Figura 5). Brinda una retroalimentación permitiendo identificar mejoras y el plan de su implementación, como así también, impedimentos (asociación *Input* en Figura 5).
En el evento *Grooming* se especifica el orden, la estimación y la granularidad de *Task* y de los *ProductBacklogItem*, representados por las asociaciones *GroomingRefined* en Figura 5.

2.6. Relaciones entre Roles y Eventos

Los eventos se utilizan en Scrum para crear reuniones entre todos los involucrados en el proceso, de manera regular, sin necesidad de incorporar nuevos eventos no definidos.
Durante cada *Sprint* el *ScrumMaster* es el responsable del entendimiento del proceso (asociación *ResponsibleOf* Figura 6). Siendo su función primordial asegurar que las políticas de la organización son adoptadas por el *DevelopmentTeam*. El *ScrumMaster* debe trabajar junto al *ProductOwner* y ambos tienen la responsabilidad de intermediar para lograr un entendimiento o negociación sobre algún problema sobre tiempos de trabajo o agregar *ProductBacklogItem* al *Sprint*, entre otros.
El *ScrumMaster* no es un Project Manager, él debe entrenar al *DevelopmentTeam* (relación *Coach* en Figura 6), esto es dirigir en el proceso Scrum. El *ScrumMaster* no tiene ninguna autoridad gerencial sobre el *DevelopmentTeam*, todos en la organización deben entender esto. Un punto crítico en Scrum es que el equipo se auto organiza para planear cómo desarrollará el trabajo dentro del *Sprint*. Durante la participación en este evento, el *DevelopmentTeam* no puede modificar su composición (relación *Participate* en Figura 6).
El *ProductOwner* es el encargado de planear, revisar y aprobar el *Sprint* (asociación *Plans* en Figura 6), en el cual es responsable de la *FirstPart* (relación *ResponsibleOf* en Figura 6). En este evento son creados, priorizados y seleccionados los *ProductBacklogItem* a ser desarrollados durante el *DevelopmentWork* (relación *Identifies* en Figura 6).
En la *SecondPart* el *DevelopmentTeam* es el responsable de diseñar el producto que desarrollará luego durante su trabajo (asociación *ResponsibleOf* en Figura 6). En esta etapa los desarrolladores tienen la

posibilidad de crear un diseño que satisfaga la solución que desean implementar.
El *DailyScrum* es realizado cada día de trabajo, a la misma hora en el mismo lugar. El *DevelopmentTeam* es el responsable de dirigirla (asociación *ResponsibleOf* en Figura 6) y el *ScrumMaster* debe asegurase que sea llevada a cabo todos los días (relación *Maintain* en Figura 6).
El *DevelopmentTeam* es el único responsable de la realización del *DevelopmentWork* (relación *ResponsibleOf* en Figura 6), toma los requerimientos, los completa y satisface, creando un producto de software. Está encargado de realizar las tareas de análisis, diseño, desarrollo, testing, documentación y mantenimiento. El equipo decide que será realizado y por qué miembro del equipo.

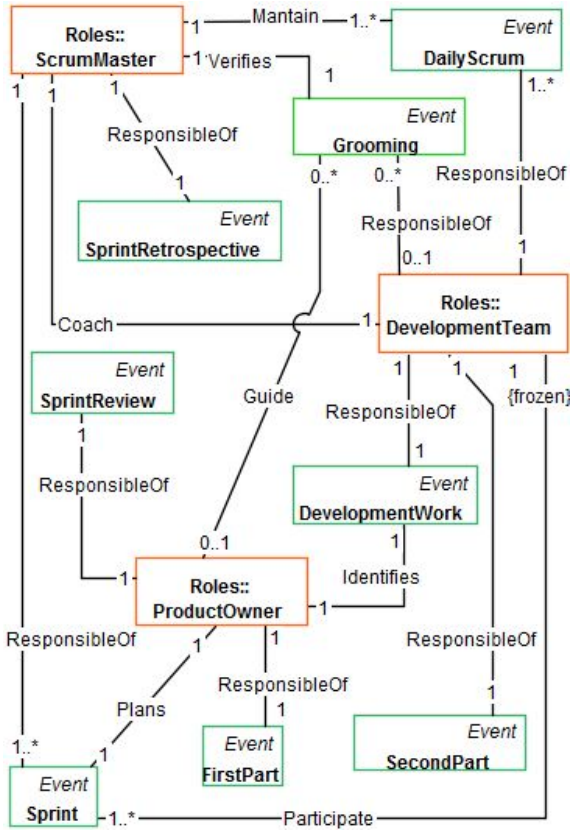

Figura 6. Representación de los Roles y Eventos.

El responsable de realizar la actividad de *Grooming* es el *DevelopmentTeam* (asociación *ResponsibleOf* en Figura 6). A su vez el *ProductOwner* está encargado de guiar el desarrollo de esta actividad (relación *Guide* en Figura 6). Al mismo tiempo, este último, debe mantener actualizado y ordenado el *ProductBacklog*. Para luego volver a estimar, priorizar y ordenarlo. El *ScrumMaster* está encargado de verificar que el *Grooming* se esté llevando a cabo (asociación *Verifies* en Figura 6).
Durante el *SprintReview* el *ProductOwner* es el responsable de revisar y aprobar el trabajo realizado durante el *Sprint* (relación *ResponsibleOf* en Figura 6). Debiendo evaluar si lo desarrollado satisface, o no, lo planeado para el *Sprint*.
También el *ScrumMaster* es responsable de la *SprintRetrospective*, esta es la oportunidad formal para enfocarse en la inspección y adaptación del proceso (asociación *ResponsibleOf* en Figura 6). Durante esta reunión el *ScrumTeam* tiene como objetivo revisar cómo fue realizado el último *Sprint*, los *Roles*, *Eventos*, y *Artefactos* que lo componen (relación *Inspect* en Figura 5). Además en ella se crea un plan de mejoras que serán aplicadas durante el siguiente *Sprint*.

2.7. Prácticas de Trazabilidad

Con el objetivo de asistir en las tareas de trazabilidad en proyectos ágiles se seleccionó un conjunto de prácticas de trazabilidad, propuestas por Jacobsson [14] y Cleland-Huang [15], que permitieron responder las preguntas de competencia. Se resume a continuación el aporte de las prácticas seleccionadas:

– Stakeholders-Requerimientos: la aplicación de esta práctica se realiza registrando el rol responsable de ingresar un *ProductBacklogItem*. Es importante que la traza sea en ambas direcciones, es decir de requerimientos a Stakeholder y de Stakeholder a requerimientos.

– Descripción inicial del problema-Requerimientos: durante el Sprint son ingresados requerimientos al *ProductBacklog*. El *ProductOwner* prioriza los requerimientos y luego, el *DevelopmentTeam* estima cuáles de estos pueden ser desarrolladas durante el Sprint en curso. Esto permite validar si el

problema inicial está resuelto y si es posible validar que todas las tareas fueron formalizadas en el *ProductBacklog*.

- Requerimientos-ProductBacklog: se agrega un identificador a los *ProductBacklogItem* y a las *Task*, haciendo posible trazar desde el *ProductBacklog* al requerimiento original.
- Requerimientos-Requerimientos: ayuda a encontrar relaciones entre diferentes *ProductBacklogItem*, permitiendo especificar cuan detallada es la información que se necesita. La información puede ser recolectada durante el *Grooming* mientras son analizados los *ProductBacklogItem* y las *Task*.
- Requerimientos-Versión: consiste en el registro de los *ProductBacklogItem*, que fueron implementados en una revisión del *Product*. Esto permite visualizar en qué se diferencian dos *version* de *Product*.
- Revisión-Versión: permite documentar la información del *ProductIncrement* evaluado durante un *SprintReview*. Se identifica el rol responsable del registro y la *version* del *Product* a la que pertenece.

3. Caso de Estudio

El modelo presentado en la Sección 2 se lo aplica a un proyecto de sistema de comercio electrónico. Durante el *SprintPlanning* se da origen a los *ProductBacklogItem*. El *ProductOwner* presenta los *ProductBacklogItem* que considera que deben ser estimados. Particularmente, durante el *FirsPart*, son concretizados como *ProductBacklogItem* los requerimientos, deseos y expectativas del cliente. En la *SecondPart*, son seleccionados los *ProductBacklogItem* a desarrollar durante el *Sprint* en curso.

De esta manera, en el evento *FirstPart* 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 Figura 7 se ilustra el *ProductBacklog* luego del refinamiento del ítem *Checkout* en los ítems: *Selección Formas de Pago y Selección Medio Envío.*

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*. La Figura 8 presenta el *SprintBacklog* de la primer iteración junto a información de los *ProductBacklogItem* y *Task* que lo componen.

Continuando con el proceso, en el *DailyScrum* y en el *DevelopmentWork* estos *ProductBacklogItem* a desarrollar son compuestos por *Task*. Estas deberán ser caracterizadas con status *Done* antes del próximo *DailyScrum*. En caso contrario el *ScrumTeam* divide los ítems del backlog (*ProductBacklogItem*) en ítems más pequeños (*Grooming*), y redimensiona el *ProductBacklog* con los nuevos elementos.

	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
Checkout	Seleccion Medio Envío	S-01013		High
	Implementación del carro	S-01008		High
	Administrar Cuenta de Usuario	S-01002		Low
	Sprint3			
	Administrar Pedidos	S-01007		Medium
	Ingreso al Sistema	S-01001		Low
	Administrar Usuarios	S-01005		Low
	Diseño Backend	S-01011		Low

Figura 7. *ProductBacklog* del caso de estudio.

También durante el *DailyScrum*, se reúne el *DevelopmentTeam* y cada *Developer* explica el progreso del *DevelopmentWork* en el *Sprint1*, comentando qué es lo que realizó desde el *DailyScrum* anterior hasta el actual, que es lo que realizará hasta el próximo y cuáles fueron sus obstáculos.

Un *DevelopmentTeam* inicia la tarea de diseñar *Header, Body, Footer*, modificando el *status* de la misma de *ToDo* a *InProgress* (Figura 8). Así mismo, si existe un problema para realizar una tarea, lo añade al *ImpedimentBacklog* y lo reporta al *ScrumMaster*. Un impedimento podría ser que el *DevelopmentTeam* está a la espera de la imagen del logo del cliente que irá en el encabezado.

PBI	Remain Total Work	Title	PBI ID	Task ID	Priority	Status	Date Start	Date Finished
Diseño Frontend	50%	Diseñar Header, Body, Footer	S-10	T-1	High	In Progress	9/1/13	
		Crear el Estilo	S-10	T-2		Done	9/1/13	10/1/13
Administrar Productos	100%	Crear Productos	S-4	T-3	High	In Progress	10/1/13	
		Actualizar Productos	S-4	T-4		ToDo		
		Eliminar Productos	S-4	T-5		ToDo		
Diseñar Base Datos	100%	Crear Tablas y Atributos	S-12	T-6	Medium	In Progress	10/1/13	

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

Una vez concluido el *Sprint1*, el *DevelopmentTeam* entrega los *ProductBacklogItem* con *status Done* al *ProductOwner*. Este último será 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* da inicio al *Sprint2*.

En otras palabras y a modo de análisis, la especificación en el caso de estudio, da como resultado que: durante el *SprintPlanning* son creados los *ProductBacklogItem*: *Diseño Frontend*, *Administrar Productos*, *Diseño Base de Datos*, *Checkout*, *Implementación del Carro*, entre otros. Una vez completados en el *DevelopmentWork* son caracterizados con *Status Done*. No obstante, también durante el *DailyScrum* y el *DevelopmentWork* puede ocurrir una redefinición de un ítem, en este caso *Checkout* es refinado en dos ítems nuevos: *Selección Formas de Pago* y *Selección Medio Envío*.

Los requerimientos que guiaron la generación del *ProductBacklogItem Checkout* son: el cliente solicita que eCommerce brinde la funcionalidad de una pantalla donde el usuario del sistema pueda acceder y seleccionar una forma de pago del producto. En ella también, el usuario debe poder seleccionar el medio de envío que desee para recibir los productos previamente seleccionados. Por consiguiente el *DevelopmentTeam* decide refinar *Checkout* en dos nuevos *ProductBacklogItem*.

Tomando en cuenta el evento *Sprint1* todos los participantes están involucrados de alguna manera. Esto se debe a que el *Sprint* es el contenedor de los demás eventos o reuniones dentro del proceso Scrum. Las funciones de cada participante en este evento a las siguientes:

ScrumTeam es el encargado de la planear *Sprint1* una etapa decide la fecha de inicio el “09/01/13”. El objetivo a cumplir es “Desarrollar la interfaz de usuario, implementar funcionalidades ABM (alta, baja y modificación) de productos, Diseñar e implementar la base de datos”. De igual forma finaliza la etapa como máximo el “30/01/13” cuando el *ProductIncrement* tenga *status Accepted*.

ScrumMaster es el responsable del *Sprint1* asegurando que las políticas de la organización, la duración y obligatoriedad de cada reunión, como así también del aprendizaje de Scrum por parte de los participantes. La duración máxima para el *SprintPlanning* será de 6 horas, para el *SprintReview* de 3 horas y para el *SprintRetrospective* 2 horas 15 minutos. Dando por iniciada el *Sprint2* una vez finalizada la *SprintRetrospective*.

ProductOwner define que en el *Sprint1* se desarrollaran los *ProductBacklogItem Diseño Frontend*, *Administrar Productos* y *Diseño Base de Datos*. Una vez finalizados (*status Done*) se revisa el *ProductIncrement1* creado satisfaga el *SprintGoal*.

DevelopmentTeam se auto organiza para desarrollar los *ProductBacklogItem*

seleccionados, cumplir el *SprintGoal* y al finalizar el *Sprint1* poder liberar el *e-Commerce versión 0.1*.

4. Conclusión

En este trabajo se propuso un modelo conceptual de Scrum representado por una serie de diferentes perspectivas de todo el framework. Así mismo 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.

Actualmente en Scrum algunas prácticas pueden ser interpretadas como trazabilidad pero sin que exista documento alguno que la registre. Una de estas prácticas puede ser una *DailyScrum* donde esta reunión le permite a los desarrolladores encontrar una manera de reflejar lo realizado durante la reunión anterior y que desarrollará en la siguiente.

Los trabajos futuros utilizarán el modelo conceptual propuesto para formalizar el soporte a tales tareas. Pudiendo añadir herramientas y técnicas de trazabilidad automáticas reduciendo la carga de trabajo para el equipo. De esta manera permitirá reducir el tiempo que deberá invertir para guardar y procesar la información de trazabilidad.

Agradecimientos

Este trabajo ha sido financiado en forma conjunta por CONICET, la Universidad Tecnológica Nacional, la Universidad Nacional de La Rioja y la Agencia Nacional de Promoción Científica y Tecnológica. Se agradece el apoyo brindado por estas instituciones.

Referencias

1. Espinoza, A., Garbajosa, J.: A study to support agile methods more effectively through traceability. Innovations in Systems and Software Engineering 7, 53–69 (2011)
2. 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)
3. IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990 1–84 (1990)
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. Jacobsson, M.: Implementing traceability in agile software development, Faculty of Engineering, LTH. Department of Computer Science. (2009).
15. Cleland-Huang, J.: Traceability in Agile Projects. Software and Systems Traceability, Springer, 265-276. (2012).

Datos de Contacto

Nazareno, Roberto. INGAR (CONICET – UTN). UNLaR, Universidad Nacional de La Rioja. Avellaneda 3657, Santa Fe, Argentina, CP: 3000. E-Mail: rnazareno@santafe-conicet.gob.ar

Leone, Horacio. INGAR (CONICET – UTN). Universidad Tecnológica Nacional, Facultad Regional Santa Fe. Avellaneda 3657, Santa Fe, Argentina. CP: 3000. E-Mail: hleone@santafe-conicet.gob.ar

Gonnet, Silvio. INGAR (CONICET – UTN). Universidad Tecnológica Nacional, Facultad Regional Santa Fe. Avellaneda 3657, Santa Fe, Argentina. CP: 3000. E-Mail: sgonnet@santafe-conicet.gob.ar