El desafío de ser un Product Owner
Responsabilidades del PO en los Proyectos Ágiles

Marcelo Estayno [1] and Judith Meles [2]

[1] Universidad Tecnológica Nacional - Facultad Regional Buenos Aires
Buenos Aires, Argentina mestayno@gmail.com
[2] Universidad Tecnológica Nacional- Facultad Regional Córdoba y Villa María
Córdoba, Argentina jmeles@gmail.com

Abstract

In Agile methodologies, Scrum in particular, the rol of the Product Owner (PO) is often assumed by a person with solid knowledge about users, market, competitors and future trends for the domain or the type of system is being developed. The PO´s challenge is to build a product vision and then place it into user stories that will help you convey that vision to the development team that will build the software.

This paper presents the problems related to the creation, prioritization, validation and acceptance of the product obtained from the user stories. This requires expertise, generic and specific skills, for which the PO is not always prepared.

Resumen

En las metodologías ágiles, Scrum en particular, el rol del Product Owner (PO) es asumido frecuentemente por una persona con conocimientos sólidos respecto de los usuarios, el mercado, la competencia y las tendencias de futuro para el dominio o el tipo de sistema que se está desarrollando. El desafío del PO es construir una visión del producto y luego plasmarla en historias de usuario (user stories), que le ayudarán a transmitir esa visión al equipo de desarrollo que va a construir el software.

En este trabajo se plantea la problemática vinculada a la creación, priorización, validación y aceptación del producto obtenido a partir de las user stories; para lo cual se requieren conocimientos técnicos para los que el PO no siempre está preparado.

Keywords— agile projetcs, , resnsabilities , user stories, skills, roles

1 INTRODUCCIÓN

La familia de métodos de desarrollo ágiles evolucionó a partir de los conocidos ciclos de vida iterativos e incrementales. Surgieron de la creencia que un acercamiento más en contacto con la realidad social y la realidad del desarrollo de productos basados en el aprendizaje, la innovación y el cambio daría mejores resultados.

El propósito principal de estas metodologías es lograr un equilibrio entre los procesos disciplinados que proponen una excesiva carga de trabajo y burocracia, y la no existencia de proceso, que lleva a desarrollar software de manera caótica.

La alternativa que ofrecen las metodologías ágiles, propone un cambio radical que hace énfasis en un replanteo de aspectos culturales, comunicacionales y sociales, que son los principales causantes de fracasos en los proyectos de desarrollo de software.

El manifiesto ágil [1], refleja el acuerdo de la mayoría de los referentes de la industria que plantean prácticas ágiles. Este manifiesto, en su declaración contiene valores y principios a los que quienes adhieren a él, se comprometen a respetar.

Este manifiesto gira en torno de privilegiar lo que es de valor para el cliente, es decir, *entregas frecuentes de software funcionando*. Propone un ambiente de colaboración con el cliente/usuario y entre los miembros del equipo y una actitud de aceptación frente a los cambios que surjan durante el desarrollo del producto de software. En este sentido se enfatiza en dar más importancia a las personas involucradas en el desarrollo de software que a las herramientas y procesos que se utilizarán para construirlo.

También se destaca la importancia y la incidencia que tiene la presencia y colaboración del cliente para alcanzar el éxito del proyecto en lugar de las extensas e infructuosas negociaciones defendiendo un contrato.

La propuesta más desafiante del manifiesto ágil tiene que ver con evaluar cada acción, cada decisión, cada elección con la vara de entregar "valor de negocio" a quién nos pide el software, que a su vez se siente parte del proyecto.

Existen actualmente un gran número de métodos ágiles, que suscriben al manifiesto. Algunos de estos métodos definen prácticas para desarrollo ágil, otros para modelado ágil y otros para gestión de proyectos ágil. En este trabajo se tomará como referencia a *SCRUM*.

Scrum es un entorno de trabajo que define prácticas para la gestión de proyectos. Si bien puede utilizarse para proyectos de propósitos diversos, se hará énfasis en proyectos de desarrollo de software.

Scrum propone un ciclo de vida iterativo e incremental, que utiliza para liberar versiones frecuentes del software, acotadas en funcionalidad, al final de cada iteración a la que llama "***Sprint***". Cada sprint es un periodo fijo de tiempo, con una duración que oscila entre dos y cuatro semanas, encuadrado en la característica de "*timeboxing*", lo que significa que la duración en tiempo es fija.

Si bien la propuesta es que al final de cada *sprint* se obtenga un producto potencialmente entregable, o sea software en condiciones de ser instalado y funcionar, muchas organizaciones eligen no liberar nueva funcionalidad del producto al fin de cada sprint. En lugar de eso, prefieren combinar el resultado de varios *Sprints* en un único entregable a liberar (*release*).

Seguidamente se describe a modo de paneo la propuesta de Scrum. El framework está compuesto por tres roles, tres entregables y cuatro reuniones o ceremonias.

Los roles definidos, que constituyen el Equipo Scrum son:

- ***Scrum Master***: es el líder de servicio cuyas responsabilidades más importantes es velar por el cumplimiento de las reglas de Scrum, como así también, dar soporte al resto del equipo y actuar como facilitador neutral preocupado en fomentar un ambiente colaborativo y en guiar al equipo hacia el mejoramiento continuo y la autosuficiencia.

- ***Product Owner***: el dueño del producto, la voz principal del cliente, quien asume la responsabilidad por su construcción y por maximizar el valor de negocio. Respecto de este rol, que es objeto principal de este trabajo, se ampliará en secciones posteriores.

- ***Development Team***: el equipo de desarrollo es el responsable de construir el producto definido por el *Product Owner*. El equipo en *Scrum* es "multi-funcional" – tiene todas las competencias y habilidades necesarias para liberar un producto potencialmente entregable al final de cada iteración. Es un equipo "auto-organizado" (auto-gestionado), con un alto grado de autonomía y responsabilidad. En *Scrum*, los equipos se auto-organizan en vez de ser dirigidos por un Líder de Proyecto.

Propone la generación de tres entregables:

- ***Product Backlog***: es la lista de características que se espera tenga el producto de software. Estas características están priorizadas y pueden expresarse a distintos niveles de granularidad. Es importante destacar que es una lista dinámica que está en constante evolución y refinamiento.

- ***Sprint Backlog***: Lista de elementos, tomada de la parte superior del *product Backlog*, que contiene la porción de características que el equipo acordó que puede construir durante una iteración (*Sprint*). Estas características deben estar priorizadas por el *Product Owner* y estimadas por el equipo.

- **Incremento del Producto**: versión del software, potencialmente desplegable en el ambiente de producción, obtenida como resultado de la ejecución de un *sprint*.

Propone cuatro tipos de eventos o ceremonias:

- ***Sprint Planning Meeting***; reunión de planificación del *sprint*: es la primera ceremonia, cuyo propósito es obtener la definición del incremento del producto que se liberará en ese s*print*. En la primera parte de la reunión, la reunión del *qué*, se define el objetivo del *sprint* y el *Sprint Backlog*. En la segunda parte de la reunión, la reunión del *cómo*, el equipo estima las características y puede descomponerlas en las tareas que necesitará hacer para desarrollarlas.

- ***Daily Meeting*:** en esta reunión se une al equipo, de pie en lo posible. Todos deben asistir, la duración aproximada recomendada es de 15 minutos. El propósito fundamental es dar visibilidad y sincronizar diariamente al equipo. Cada integrante a su turno debe responder tres preguntas: *¿Qué hizo desde la reunión anterior? ¿Qué piensa hacer hasta la próxima reunión? ¿Qué impedimentos se le presentaron?*

- ***Sprint Review Meeting***: la reunión de revisión, también conocida como "*Sprint Demo*", es el momento en el cual el equipo le presenta al *Product Owner* el incremento del producto que obtuvo durante la ejecución del s*print*. Se hace al final del *sprint*, su propósito es obtener realimentación sobre el producto. El *Product Owner* evaluará las funcionalidades desarrolladas y determinará cuales va a aceptar y cuáles no.

- ***Sprint Retrospective Meeting***: Está es la última reunión que se realiza en un Sprint, el propósito es obtener realimentación del proceso utilizado, repasar aspectos positivos y negativos y definir acciones a seguir para mejorar la calidad del proceso.

Si bien *Scrum* propone 4 ceremonias, hay una quinta ceremonia, que a pesar de no estar explícitamente definida en el framework, es muy importante y es necesaria. En este evento denominado *Grooming*, el *Product Owner* tiene un rol fundamental, y es la reunión de creación, preparación y mantenimiento del *Product Backlog*. Esta reunión no tiene un momento fijo predefinido de realización, no obstante, aconsejamos realizarla en las siguientes oportunidades: para definir el *Product Backlog* por primera vez, antes de cada *Sprint Planning* y en cualquier momento que el equipo de desarrollo considere necesario, durante la ejecución de un Sprint, para comprender y refinar los requisitos que se están desarrollando.

Scrum propone un nivel bajo de ceremonia en el proceso, lo que significa que son pocas prácticas las que define, no obstante no debemos confundir ágil con débil o permisivo, dado que el método es estricto en el cumplimiento de sus definiciones y sus reglas.

Por ser *Scrum* un framework de "gestión", no define lineamiento alguno respecto a cuestiones vinculadas a la ingeniería del producto, ni da detalle sobre cómo construir el software, asumiendo que el equipo de desarrollo sabe, está capacitado y en condiciones, para resolver esa misión.

Las diferencias que pueden destacarse de este enfoque de gestión ágil respecto de la gestión tradicional de proyectos están sustentadas fuertemente en el compromiso, la auto-organización, la autogestión y el esfuerzo colaborativo. Estas competencias, y aptitudes deben desarrollarse en las personas para obtener los resultados esperados.

Al centrar en nuestro caso la atención en el rol del *Product Owner* y en el conjunto de responsabilidades asignadas a él, surgen los interrogantes respecto a si quién asuma este rol realmente sabrá ¿cómo conectar el negocio con el desarrollo del producto de software? ¿Cómo plasmar las estrategias y el retorno de inversión en las user stories? ¿Cómo priorizar user stories para maximizar el valor de negocio? Estos son algunos de los interrogantes, y este es el desafío.

2 DEFINICIÓN DEL PERFIL DE PRODUCT OWNER

Mayer en su reciente libro [2] denomina al *Product Owner* (PO) como *"la voz del QUE"*, una voz única, posiblemente canalizando muchas voces, con el propósito de definir resultados "bien formados" y priorizar el valor más alto posible en un contexto dado. La voz del "qué" también es responsable de los gastos y de la decisión de continuar o no.

El perfil del Product Owner reúne varias aptitudes. El PO *como agente de cambio*, actúa como catalizador del cambio en la organización, posibilitando la creación de valor a través de proyectos y productos. El PO crea una conexión entre cómo debe verse el negocio en el futuro y su estado actual.

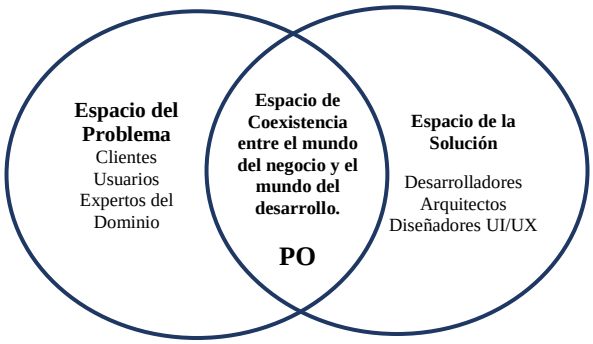
El PO es un *facilitador* clave en la organización, para unir al cliente y la comunidad de negocio con el equipo de desarrollo ágil.

El PO tiene un desafío en su trabajo, pues debe *crear la confianza* necesaria para que el flujo de información fluya entre los mundos que intenta vincular. Actuando como *interfaz entre los involucrados*, en muchos casos deberá *definir acuerdos* entre intereses conflictivos. También debe *colaborar* continuamente y *proveer retroalimentación* permanente, en ambos sentidos.

Dentro del conjunto de competencias que debe desarrollar un PO está la de *comunicador*. La comunicación es el principal vehículo facilitador del aprendizaje y para que el equipo pueda lograr una comunicación efectiva, es vital que el PO tenga presente los filtros existentes en todos los individuos al momento de hacer efectiva esa comunicación. Estos filtros son los valores personales, las creencias, los intereses, las expectativas y también experiencias vividas con anterioridad. [3] Cuando comunicamos somos subjetivos y nuestra tendencia es alejarnos de mensajes que entran en conflicto con nuestras ideas y creencias. Tendemos a oír sólo lo que queremos oír, ponemos más atención a aquellas cosas que más nos interesan.

En este contexto y a modo de síntesis se presenta el gráfico de la figura 1, que muestra al PO como un "*puente" o "vínculo"* entre el espacio del problema y espacio de la solución, ubicado en lo que denominamos espacio de coexistencia. El PO debe comprender ambos mundos, el del negocio y el del desarrollo, y debe hacer de traductor entre estos dos mundos. El PO debe tener la perspectiva general del producto final.

Figura 1: Product Owner (PO) con su espacio de interconexión



3 DESCRIPCION DE RESPONSABILIDADES DEL PRODUCT OWNER

Luego de un análisis realizado a las propuestas de Schwaber [4], Schwaber [5], Rubin [6] y Nir [3], en esta sección se resume el conjunto de responsabilidades asignadas a un Product Owner en el contexto de un proyecto ágil.

A. Definir el producto

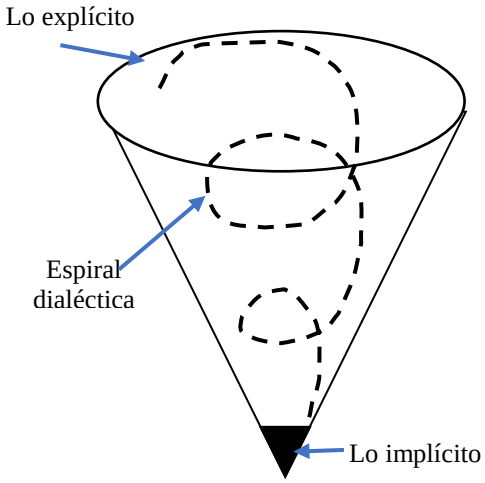
La elicitación permitirá la definición del producto, es un proceso de comunicación analítica y de flujo libre.
La elicitación trata de *comprender las necesidades* más que de capturar deseos. Es un esfuerzo colaborativo que se ajusta bien al desarrollo propuesto por el Manifiesto Ágil [1] y ocurre en la ceremonia que se describió antes como Grooming.

Este es un trabajo de descubrimiento, no una mera recolección pasiva, que asume que el producto ya está definido y sólo hay que plasmarlo. Muy por el contrario, la creación se produce atravesando lo que Pichon Rivière [7] llama "*espiral dialéctica*".

Tal como se grafica en la figura 2 se lo representa como un cono invertido con una base, un vértice y la espiral dialéctica. En la base: Se ubican los contenidos emergentes, manifiestos, o "explícitos". En el vértice: Las situaciones básicas o universales "implícitas". La espiral grafica el movimiento dialéctico de indagación y esclarecimiento que va de lo explícito a lo implícito, con el objeto de explicitarlo. Hacer explícito lo implícito es lo que concretamente se llama "*interpretación"*. Esta responsabilidad puede considerarse una de las principales atribuida al rol de PO.

Estas responsabilidades en el contexto de los procesos definidos recaían entre las tareas que debía realizar el rol conocido como Analista Funcional o Analista de Negocios.

Figura 2: Espiral dialéctica: analizar es hacer explícito lo implícito.



Ahora bien, ¿qué proponen la mayoría de las metodologías ágiles, en la actualidad como herramienta facilitadora de este proceso dialéctico? Proponen el uso de "*user stories*".

Las user stories no requieren de una especificación exhaustiva de la solución "*up-front*", en lugar de eso estimulan un diálogo fluido y enriquecedor entre los involucrados, durante la construcción del software, buscando la mejor solución.

Según Albert Mehrabian [8], la comunicación humana se compone de tres partes:

1. En un 7%: El contenido (las palabras, lo dicho)

2. En un 38%: El tono de la voz

3. En un 55%: Las expresiones faciales y corporales

De aquí el fundamento de uno de los principios del manifiesto ágil de privilegiar el contacto cara-a-cara entre los involucrados, para poder lograr una comunicación sólida, completa y significativa.

Las *user stories* reflejan una necesidad del usuario, una descripción del producto, un mecanismo para diferir una conversación, como así también un ítem de planificación.

El modelo de las 3C´s, propuesto por Ron Jeffries [9] para distinguir user stories "sociales" de las prácticas de requerimientos "documentales", plantea estos tres componentes:

- **Card (Tarjeta)**: Es la manifestación más visible de la user story, pero no la más importante, debe "plasmar" el requerimiento, no "especificarlo" La user story debe poder escribirse en una tarjeta o post-it de tamaño estándar. Si no alcanza el espacio será una señal de que estamos traspasando las fronteras y comunicando demasiada información que debería compartirse cara a cara. La sintaxis sugerida para expresar las User Stories nos permite responder a las preguntas:
  - ✓ *quien*, representa el rol que necesita la funcionalidad;
  - ✓ *qué*, permite la segmentación de la funcionalidad del producto;
  - ✓ *porqué*, comunica la utilidad de la funcionalidad desde la perspectiva del negocio.

Por lo tanto la sintaxis se expresa de la siguiente forma:

Como <nombre del rol>, yo puedo <actividad> de forma tal que <valor de negocio que recibo> [10]

La figura 3 muestra un ejemplo de una *user story* para el software de un GPS.

Figura 3: Ejemplo de la tarjeta (Card) de una user story con la sintaxis sugerida

- **Conversación**: Toda *user story* debe tener una conversación con el *Product Owner*. Una comunicación cara a cara que intercambia no sólo información sino también experiencias, pensamientos, opiniones y sentimientos. Son la parte más importante de la *user story*. El detalle que se obtiene de la conversación debe ser un resultado de esta conversación, no un reemplazo de la misma.
- **Confirmación** – Sirve para determinar lo que el *Product Owner* espera, es decir para que el equipo de desarrollo sepa lo que debe construir. Son pruebas que comunican y documentan detalles y que se pueden utilizar para determinar si una historia está completa.

Durante el proceso de descubrimiento y definición del producto, puede surgir la necesidad de trabajar con diversos niveles de granularidad que están estrechamente vinculados con el tamaño y el nivel de detalle de lo que definimos. Surge así una jerarquía que facilita el trabajo con los requerimientos.

A continuación se presenta la jerarquía propuesta por Rubin [6], que es coincidente con la presentada por Leffingwell [11]. Ellos plantean la *Epic* (Épica), para indicar historias que necesitan de varios meses para desarrollarse. El término alude a la idea de historias como "El señor de los anillos" o "La guerra y la paz". Las épicas son

marcadores que contienen una gran colección de historias que serán detalladas en forma posterior.

El siguiente nivel de historia es el que tiene un tamaño que excede la duración del sprint, y suele llamarse *Feature* (Característica). Finalmente, la forma más pequeña, es la *user story*, que suele llamarse también historia implementable, porque puede desarrollarse en forma completa en un sprint.

Finalmente, se llama *Theme* (Tema) a la colección de user stories relacionadas. Los temas proveen una forma conveniente de agrupar user stories que tienen algo en común.

Trabajar con niveles de granularidad diferentes para los requerimientos reduce el nivel de especificaciones demasiado tempranas y la necesidad de tomar decisiones cuando no hay suficiente información para hacerlo, además disminuye la sobrecarga de administración para productos grandes.

Escribir los requerimientos como *user stories* es trabajo del PO, quién de ninguna manera debe limitarse simplemente a leer sus historias, sino por el contrario debe preguntarle a los miembros del equipo de desarrollo, asegurarse que entendieron. Si es necesario deberá utilizar cualquier recurso que considere para garantizar la comprensión, como por ejemplo mapas conceptuales, gráficos, dibujos.

La priorización es una asignación destacada dentro del conjunto de responsabilidades, el PO debe comprender las necesidades de los involucrados y conocer el negocio para poder comunicar al equipo de desarrollo qué construir y en qué orden.

Como consecuencia de la actividad de priorización, se gestiona el *product backlog* organizando los ítems de mayor importancia en la parte superior de la pila y los de menor importancia en la parte inferior. Del mismo modo, los elementos priorizados y próximos a ingresar al sprint backlog deben estar preparados y con el nivel de granularidad de una *user story*, mientras que los elementos de menor prioridad pueden definirse con un nivel de granularidad mayor en la forma de épicas o temas.

B. Participar en la Planificación

El PO es un participante clave en todos los niveles de planificación *[3]* presentes en una organización. El nivel más global, conocido como *nivel de portfolio*. A este nivel se determina en qué productos de software trabajar y en qué orden, pero a largo plazo.

El siguiente nivel es la planificación del producto, conocida también como *conceptualización o envisioning*, es la actividad donde se captura la esencia y se crea un plan aproximado para la creación del producto. Comienza con la creación de la visión del producto, seguido por la creación de la definición a alto nivel del product backlog.

Luego continúa la planificación a nivel de release, cuyo propósito es comunicar con la precisión que es razonablemente posible cuando se entregará la versión del producto, que características incluirá, cuánto costará.

Por último, la planificación de más corto plazo, que es la planificación del sprint, durante la cual el PO trabaja con el equipo de desarrollo para seleccionar un conjunto de ítems de *product backlog* que el equipo realistamente pueda entregar al final del s*print*.

En este caso las responsabilidades antes descriptas, en el contexto del desarrollo de software “tradicional”, recaían en los roles de Administrador o Líder de Proyecto, Administrador de Producto y/o Administrador de Programa. Las organizaciones han invertido recursos en la formación de las personas que se desempañarían en estos roles.

C. Definir Criterios de Aceptación

Los criterios de aceptación son condiciones de satisfacción, específicos para cada user story bajos las cuales el producto debe satisfacer los requerimientos funcionales y no funcionales[1]. El PO debe asegurarse que los criterios de aceptación sean especificados y que sean construidas y ejecutadas las pruebas que verificarán el cumplimiento de esos criterios en el producto. En estos aspectos el PO actúa en parte como analista de negocio y en parte como analista de pruebas.
Sin estos criterios, el equipo tendría una comprensión incompleta del ítem del *product backlog*. Por esta razón, muchos equipos incluyen la existencia clara de criterios de aceptación como parte de la definición del criterio de listo[2].

D. Gestionar la Economía

El PO es responsable de asegurar que consistentemente se tomen buenas decisiones económicas relacionadas con el *product Backlog, los Sprints y los Releases.*

A nivel de *release* el PO debe realizar compensaciones permanentes respecto del alcance, presupuesto, fechas y calidad, conforme recibe información que fluye durante el desarrollo del producto. Las compensaciones realizadas el principio del release, pueden no ser las adecuadas frente a la presencia de nueva información que llega durante la ejecución del mismo.

El PO, al revisar los ítems pendientes del *product backlog*, podrá decidir que el costo de crearlos es mayor que el valor que entregarán y como consecuencia removerá esos ítems.

También puede decidir cambiar la cadencia de entrega, que por ejemplo estaba definida cada 5 *Sprints*, liberando versiones del producto al final de cada sprint.

A nivel de cada *sprint*, el PO también gestionará la economía, asegurándose que se entregue un buen retorno de inversión (ROI) al final de cada sprint.

Respecto de la visión de la economía sobre el *product backlog*, dado que el PO es quien lo prioriza, si las condiciones económicas cambian, las prioridades del *product backlog* cambiarán también.

E. Aceptar el Producto

Las pruebas de aceptación tienen por objeto demostrar que una aplicación es aceptable para el *Product Owner* que ha sido responsable de guiar el desarrollo del sistema, Leffingwell [11]. Esto significa que es él quien debería ejecutar las pruebas de aceptación y confirmar que se han alcanzado los criterios de aceptación.

Ahora bien, el PO puede decidir correr las pruebas de aceptación él mismo, o requerir la asistencia de usuarios expertos que ayuden a confirmar que la funcionalidad cumple las condiciones de satisfacción. No obstante ello, el PO es quien debe dar la mirada final respecto de si los ítems del *product backlog* desarrollados, satisfacen sus expectativas.

Para sintetizar, la mayoría del trabajo realizado por el PO puede resumirse en un sentido amplio como *la creación e incremento de valor y la reducción y eliminación de costos para el negocio*.

---

[1] **Requerimientos funcionales**: referidos al comportamiento esperado para el software, comúnmente referido como el "qué".
Requerimientos no funcionales referidos a las condiciones o capacidades no comportamentales, el "como" que el sistema debe satisfacer. También conocidos como requerimientos de calidad.

[2] **Criterio de Listo (*Ready criteria*)**: se aplica a las user stories y se utiliza para asegurar que están en condiciones de ser incluidas en la siguiente iteración del proyecto.

4 ¿QUIÉN ASUME EL ROL PRODUCT OWNER?

En la mayoría de las organizaciones que no trabajan con Scrum, probablemente no tengan un rol etiquetado bajo el nombre de "product owner". Por lo tanto, ¿quién podrá asumir ese rol? Para abordar este tema, previamente es conveniente identificar los diversos escenarios que pueden presentarse al momento de desarrollar software, estos escenarios representan las circunstancias en las cuales se conforman los equipos y a qué tipo de organizaciones representan.

Quién debe ser el PO depende de qué tipo de esfuerzo de desarrollo se trata y de cada organización específica. A continuación se toma el enfoque de Rubin [6] para la identificación de contextos de desarrollo para los que se determina qué persona es la más adecuada para asumir el rol.

A. *Desarrollo interno*: al desarrollar software perteneciendo a un área o gerencia interna de una empresa, la persona adecuada para asumir el rol de PO debe ser quién se beneficiará con el desarrollo, es decir un representante del área de negocio. Por ejemplo si el producto de software a desarrollar es para la gerencia de producción, una persona facultada de esa gerencia debería ser el PO. Algunas organizaciones, que aún no han aprendido la importancia de tener una persona de negocios comprometida día a día con el desarrollo del producto, pueden pedir que alguien del área de sistemas asuma las responsabilidades diarias de un PO. Dado que se sabe que esta persona de IT no está facultada para tomar decisiones importantes, las organizaciones que hacen esto, habrán cubierto el rol de manera confusa e ineficiente

B. *Desarrollo de un producto comercial "enlatado"*: en este escenario una compañía construye un producto para vender luego a clientes externos. Aquí quién asume el rol de PO es un empleado de la compañía que actúa como la voz del cliente real. Con frecuencia, esta persona viene de las filas de gestión o marketing del producto, por ejemplo un *product manager* o *project manager*.

C. *Desarrollo de componentes*: algunas compañías se dedican a la construcción de componentes que son partes de soluciones, que luego se utilizarán ensamblándolas para construir soluciones completas de valor para un cliente. El foco de desarrollo de estos equipos es a un nivel más específico, es por esto que el PO en estos casos es una persona con un perfil técnico, que es capaz de definir y priorizar estos aspectos en sus *backlogs*. De hecho es un perfil considerablemente más técnico que el de los PO de los otros escenarios.

D. *Desarrollo de un producto subcontratado*: este es el caso en el que una compañía X contrata a otra compañía para que desarrolle un producto de software. La persona que asuma el rol de PO debe ser de la compañía X, es decir de la compañía que está contratando y pagando por la solución y recibirá los beneficios. Eventualmente puede asignarse a una persona interna de la compañía contratada para que actúe como enlace y desde luego el equipo de desarrollo y el Scrum Master serán de la compañía contratada.

5 HABILIDADES ESPERADAS PARA EL PRODUCT OWNER

Lo expresado en la sección II, Definición del perfil del PO, es lo definido hasta ahora en la literatura ágil. A partir de lo investigado para este trabajo, se propone un conjunto de habilidades esperadas que complementan el perfil. Las mismas se exponen a continuación:

Convertir conocimiento tácito en explícito: El conocimiento tácito se trata del conocimiento personal o propio del individuo, este conocimiento se halla profundamente imbricado en la mente de la persona y ampliamente relacionado con la experiencia práctica de la misma. El conocimiento explícito, ese otro tipo de conocimiento, caracterizado por ser más formal y sistemático, que puede ser transmitido al equipo de desarrollo. Nonaka y Takeuchi, [12].

Manejo de Léxico Técnico: Entre otras competencias que debe tener el PO, es un léxico y comprensión de términos que seguramente son más propios del desarrollo del software que de su área de experiencia. También se requiere la competencia de integrar un equipo de trabajo con base tecnológica.

6 CONCLUSIONES

Del trabajo realizado, se desprende que se han transferido responsabilidades que originalmente se atribuían a otros roles, en los proyectos de desarrollo tradicionales, basados en procesos definidos, al rol del *Product Owner* propuesto por *Scrum*, en su gestión ágil de proyectos.

Se pueden mencionar la acción de definición, gestión y priorización del producto, que en la gestión tradicional es responsabilidad del Analista Funcional y la acción de planificación y gestión de la economía que recae en la figura del Líder o Administrador del proyecto. La acción de la definición de las pruebas de aceptación que en los proyectos tradicionales recae en un Analista de Pruebas

El PO actúa como punto focal para la gestión estratégica y táctica del producto. Es un rol estratégico para SCRUM y también puede ser su "talón de Aquiles", dado el riesgo de convertirse en el eslabón más débil de la cadena de valor del proceso de desarrollo de software. Este riesgo estará latente mientras no se reconozca la necesidad de formar a quienes asumirán este rol en los proyectos ágiles.

Claramente el rol de Product Owner es un rol de dedicación full-time, con responsabilidades significativas. De hecho uno podría preguntarse, luego de leer todas las competencias que se le atribuyen, si realmente una única persona podrá hacerlas a todas.

Al momento de elegir la persona que asumirá este rol, es importante tener en mente no sólo quien debe ser PO, sino también y muy especialmente quien puede serlo.

También es importante considerar que debe recibir un entrenamiento previo, muy exhaustivo. Este entrenamiento debe incluirse como parte de los costos y tiempos del proyecto.

7 REFERENCIAS

[1] Authors: The Agile Manifesto, "Manifesto for Agile Software Development," 2001. [Online]. Available: http://agilemanifesto.org/. [Accessed 18 Abril 2014].

[2] T. Mayer, The Peoples´s Scrum- Agile Ideas for Revolutionary Transformation, San Francisco: Dymaxicon, 2013.

[3] M. Nir, Agile Product Owner Secrets, London: Sapir Consulting, 2014.

[4] K. &. B. M. Schwaber, Agile Software Development with Scrum, London: Prentice Hall, 2001.

[5] K. &. S. J. Schwaber, "The Scrum Primer - The Definitive Guide to Scrum: The Rules of the Game," Julio 2013. [Online]. Available: https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf. [Accessed 12 07 2014].

[6] K. Rubin, Essential Scrum- A practical guide to the most popular agile process, Michigan: Addison Wesley, 2013.

[7] P. R. Enrique, El proceso grupal, Buenos Aires: Nueva Visión, 1981.

[8] A. Mehrabian, Silent messages., Oxford: Wadsworth, 1971.

[9] R. Jeffries, "XProgramming.com," 30 08 2001. [Online]. Available: http://xprogramming.com/articles/expcardconversationconfirmation/. [Accessed 25 02 2014].

[10] M. Cohn, User Stories Applied, Boston: Addison Wesley, 2004.

[11] D. Leffingwell, Agile Software Requeriments, Boston: Addison Wesley, 2011.

[12] H. Nonaka I & Takeuchi, The knowledge-creating company: How Japanese Create the Dynamics of Innovations, New York-Oxford: Oxford University Press, 1995.