Trazabilidad Ágil

Paula Izaurralde, Natalia Andriano
Universidad Tecnológica Nacional, Facultad Regional Córdoba

Abstract
El presente trabajo resume el esfuerzo de investigación desarrollado en la primera etapa dentro del marco de la tesis de maestría "Modelo Adaptable de Trazabilidad de Requerimientos de Software en Entornos Ágiles de gran escala", el cual tiene por objetivo revisar las prácticas de ingeniería de software para el desarrollo de requerimientos, con foco en el proceso de especificación de los mismos. Se plantea como objetivo determinar los principales atributos y características que debe cumplir un requerimiento, revisando la bibliografía existente sobre técnicas de especificación de requerimientos de software en Scrum e identificando las características de un buen requerimiento según IEEE. Por último, se analiza el uso y la importancia de la trazabilidad de los requerimientos en proyectos de desarrollo de software que adoptan métodos ágiles, para evaluar si la implementación de dicho criterio de calidad es útil o no. En particular se concluye que las historias de usuario son requerimientos de software, ya que expresan el problema que el sistema o producto software debe resolver y se descubre que el modelo INVEST para la definición de historias de usuario no contempla algunos de los atributos de calidad planteados por la IEEE, más precisamente los criterios de completitud y trazabilidad.

Palabras Clave: trazabilidad, requerimientos, agiles, historias de usuarios, Scrum, Modelo INVEST, IEEE

1. Introducción

Por mucho tiempo se ha pensado que la captura de los requerimientos es una fase temprana de los proyectos de desarrollo de software, que una vez completada sienta las bases a partir de las cuales se lleva a cabo el desarrollo del mismo [1]. La realidad ha demostrado que los clientes rara vez conocen sus propias necesidades con suficiente profundidad como para definirlas a priori, ya que durante la vida del proyecto, las necesidades y prioridades de los clientes cambian [1]. En este marco y en respuesta a los desafíos presentados por el desarrollo de productos de software modernos en relación a la obtención (*elicitation* en inglés) y gestión de cambios de requerimientos [2], que no han podido ser abordados por los procesos de desarrollo tradicionales, surgen las metodologías ágiles dando lugar a la creatividad y sensibilidad frente a condiciones cambiantes enfatizando la participación del cliente y la rápida reacción a los cambios de requerimientos y entregas continuas del producto [3].

Las metodologías ágiles se basan en la premisa de que en las fases iniciales de un proyecto, los requerimientos deben explorarse en un alto nivel de abstracción [4]. El objetivo de esta primera etapa consiste en comprender el alcance del proyecto, identificar los objetivos, construir una visión de desarrollo común y determinar los requerimientos iniciales [5]. Los dueños del producto mantienen una lista de requerimientos de software priorizada de acuerdo al valor de negocio que cada una de ellas provee. De esta lista se toman los principales ítems para el desarrollo de cada iteración. Los requerimientos son analizados con mayor detalle a medida que van siendo implementados en las iteraciones [4].

Para lograr verificar que dichos requerimientos son realmente implementados surge como un atributo de calidad [6] la trazabilidad de los requerimientos de software. Trazabilidad se refiere a la habilidad de poder describir y seguir la vida de un requerimiento en ambas direcciones, es decir, desde su origen y especificación, hasta su implementación y uso, como así también durante su constante refinamiento [7]. Permite determinar que todos los requerimientos han sido completamente desarrollados, abarca la

relación con otros artefactos de trabajo. Es utilizada fundamentalmente cuando es preciso evaluar el impacto de los cambios en las actividades del proyecto o en los artefactos de trabajo [8].

La trazabilidad es una práctica de ingeniería que se propone de forma independiente al proceso o metodología de desarrollo [9]. Por lo tanto, los métodos agiles no están exentos a mantener la trazabilidad de los requerimientos de software [10]. Sin embargo, existen opiniones encontradas respecto a la importancia de la trazabilidad de los requerimientos en las metodologías ágiles. Por un lado Scott W. Ambler [4] sostiene que los beneficios de poseer una matriz de trazabilidad hacen más sencillo el análisis del impacto de un cambio de requerimiento; pero también considera que poseer una o más personas familiarizadas con el sistema generan el mismo efecto, y resulta más fácil y económico solicitar a estos expertos que estimen el impacto del cambio. Las matrices de trazabilidad han sido sobrevaloradas, dado que los costos de mantenerlas, aún cuando se utilizan herramientas específicas a tal fin, superan por mucho a los beneficios [4]. Por otro lado, existen quienes creen [11] que es más preciso determinar los costos y el tiempo que llevará implementar un cambio cuando se posee una trazabilidad completa en vez de contar con un ingeniero o un programador familiarizado con todas las áreas afectadas por el mismo.

El presente trabajo muestra los resultados de la investigación realizada durante la primera etapa del plan de tesis de Maestría en Ingeniería en Sistemas de Información, “Modelo Adaptable de Trazabilidad de Requerimientos de Software en Entornos Ágiles de gran escala” UTN-FRC. Dicha etapa presenta 2 objetivos bien diferenciados: 1) Explorar y obtener conocimiento acerca de diferentes formas en las que pueden encontrarse especificados los requerimientos de software en entornos ágiles mediante la identificación de los atributos y características de un buen requerimiento y 2) Explorar y obtener conocimiento teórico sobre buenas prácticas de gestión de requerimientos desde modelos más tradicionales a modelos más “ágiles”. En este contexto la gestión de requerimientos se limita tan sólo al desarrollo y manutención de la trazabilidad de los mismos.

Se debe destacar que el primer objetivo de investigación fue presentado como trabajo de especialidad [13].

El trabajo será elaborado en diferentes secciones. La sección de Elementos de Trabajo y Metodología define cuáles son los pasos llevados a cabo durante la investigación y presenta los conceptos sobre los que se basó dicho análisis; la sección Resultados y discusiones presentan los descubrimientos de la investigación y los hallazgos realizados por varios autores. Por último, se desarrollarán las conclusiones del trabajo y detallarán los pasos a seguir en las próximas etapas de investigación.

2. Elementos del Trabajo y metodología

Para poder dar comienzo a dicho análisis, como primer paso fue preciso determinar cómo se especifican los requerimientos en metodologías ágiles como Scrum [14] y qué características deben poseer dichas especificaciones para cumplir con las características de un buen requerimiento según la IEEE [6].

2.1 Requerimientos y especificación

Para ello se tomaron como base los siguientes conceptos. Requerimiento es [15]:

(1) Una condición o capacidad requerida por un usuario para resolver un problema o alcanzar un objetivo.

(2) Una condición o capacidad que debe ser poseída por un sistema o componente de un sistema para satisfacer un contrato, un estándar, una especificación u otro tipo de documento formalmente impuesto.

(3) Una representación documentada de una condición o capacidad según 1 y 2.

La especificación de requerimientos es el paso en donde los resultados de la identificación de los requerimientos se "retratan" [16]. El proceso de especificación tiene por objetivo obtener documentación no ambigua y completa de los requerimientos de software. Los beneficios de la especificación de los requerimientos de software se describen a continuación [6]:

- Establecer una base de acuerdo entre los clientes y los proveedores acerca de lo que el software debe hacer.
- Reducir el esfuerzo de desarrollo: Dado que se especifican y validan previo al desarrollo.
- Servir de base para estimar costos y calendario.
- Sentar las bases a partir de las cuales se pueden efectuar actividades de verificación y validación.
- Facilitar la transferencia de los requerimientos de software a otros interesados.
- Servir de base para elaborar mejoras posteriores al producto.

La especificación de los requerimientos de software es el proceso de grabado o el registro de los requerimientos en una o más formas, incluyendo el lenguaje natural y formal, representaciones simbólicas o gráficas [17].

Una característica importante sobre la especificación de los requerimientos de software es que debe ser escrita por uno o más representantes del proveedor, uno o más representantes del cliente o por ambos [18].

Desde el punto de vista de las metodologías ágiles, las Historias de Usuario se focalizan en establecer conversaciones acerca de las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea dicha funcionalidad, usualmente un usuario [19].

Poseen las siguientes características: una descripción escrita que será utilizada para planificar y posteriormente disgregar los detalles con el dueño del producto, las conversaciones propiamente dichas con el dueño del producto y las pruebas que han de determinar si las historias están finalizadas o no [20]. Generalmente se las escribe en post-its (notas), y se las dispone en una pared o una mesa para facilitar de este modo la planificación y discusiones que se llevan a cabo durante la misma. La parte más importante de las historias de usuario es la conversación que se genera en torno a las mismas. Estas notas representan los requerimientos del cliente y típicamente se las escribe como se muestra en la siguiente figura:

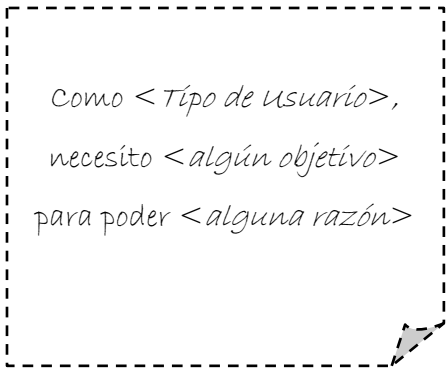

Figura 1: Historia de Usuario.

Uno de los beneficios de las Historias de Usuario es que pueden ser escritas en diferentes niveles de detalle. Es posible escribir historias que cubren múltiples funcionalidades. Estas historias más grandes son llamadas Épicas y dado que típicamente no pueden ser finalizadas en una iteración, se las divide en múltiples Historias de Usuario [21].

En la actualidad, los requerimientos de software pueden encontrarse especificados tanto en libros de requerimientos [2], como así también en historias de usuario, épicas, escenarios de prueba como el Desarrollo Guiado por Pruebas o TDD (por sus siglas

en inglés *Test Driven Development*) [22] y el Desarrollo Guiado por Comportamiento o BDD (por sus siglas en inglés *Behavior Driven Development*), entre otros [24].

2.2 Las historias de usuario en el proceso de desarrollo de requerimientos

Las Historias de Usuario son escritas a lo largo de todo el proyecto de desarrollo. Usualmente al comenzar un proyecto se lleva a cabo un *workshop* donde participan todos los miembros del equipo, con el fin de crear un *product backlog*[1] que describa las funcionalidades que van a desarrollarse en el curso de los siguientes tres a seis meses [25].

Los dueños del producto son responsables de que exista una pila de producto compuesta de Historias de Usuario, sin embargo, no necesariamente son ellos quienes deben escribirlas. Lo importante es que participen en las discusiones que aportan mayor detalle sobre las necesidades de los usuarios [25].

Uno de los problemas que la mayoría de las organizaciones tiene que enfrentar es mantener una pila de producto actualizada para que los ítems del producto puedan ser tomados por los equipos de desarrollo [26]. La definición y gestión de requerimientos ágiles ha sido diseñada específicamente para resolver este problema. El objetivo es alimentar la pila del producto a un ritmo mayor al que los equipos de desarrollo pueden generar código. Este marco de trabajo puede utilizarse tanto para la generación de requerimientos justo a tiempo o *just in time* (JIT) como para el armado de un repositorio de requerimientos que han de desarrollarse en el futuro [26]. Se utiliza un ciclo similar al ciclo Scrum [25] que emplean los equipos de desarrollo. La diferencia es que este ciclo se encuentra dos o tres etapas adelantado a los equipos de desarrollo (Ver Figura 2: Definición y Gestión de Requerimientos Ágiles [26]).

El objetivo es crear un proceso en el cual se definan, revisen, organicen y comuniquen los requerimientos. El proceso comienza identificando y construyendo una pila de requerimientos. Esta pila es una lista de elementos que deben definirse en orden para poder alimentar la pila del producto.

El objetivo final puede ser una lista de historias de usuario, requerimientos funcionales, etc. El equipo de requerimientos decide, basado en las estrategias y objetivos del negocio, qué cosas deben definirse. Al igual que los equipos de desarrollo, el equipo de requerimientos puede planear su iteración, llevar a cabo el trabajo y finalmente tener una reunión de revisión. Si los resultados de la iteración cumplen con las expectativas planteadas, entonces pueden moverse a la pila del producto. En muchos casos, las organizaciones desarrollarán documentos que no necesariamente terminarán en la pila del producto, sino que serán consultados por los equipos durante el desarrollo.

Aquí es donde la trazabilidad de los ítems del producto a cualquier otro documento externo se torna importante. Otra de las partes importantes de este marco de trabajo es la Descomposición. La Descomposición es el proceso por el cual los ítems del producto son comunicados y refinados junto con los equipos de desarrollo. En Scrum esto es conocido como preparación de la pila del producto o *backlog grooming* [26].

Cabe destacar que los mismos principios [26] que se utilizan en el desarrollo de software bajo metodologías ágiles pueden aplicarse también a la definición y gestión de requerimientos.

[1] Product Backlog se refiere a una lista priorizada de historias de usuario.

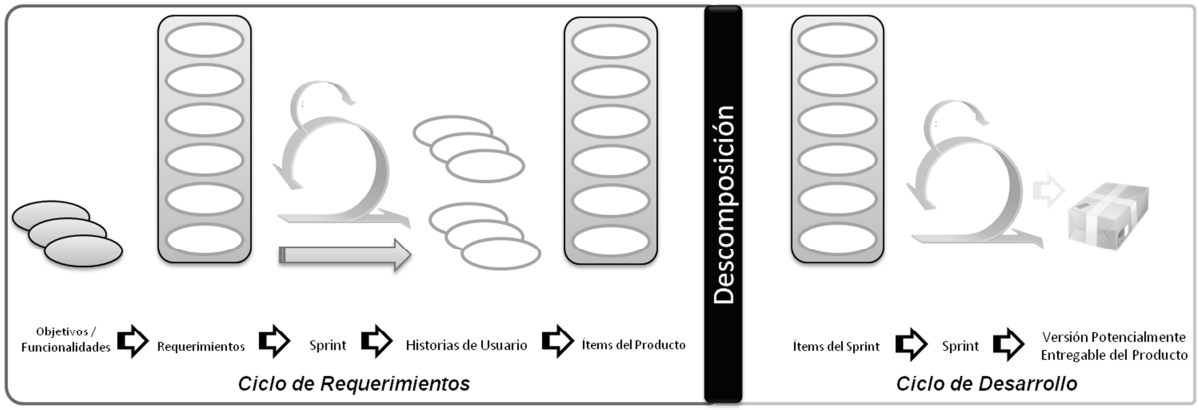

Figura 2: Definición y Gestión de Requerimientos Ágiles [26]

2.3 Trazabilidad

Como segundo paso fue necesario entender qué es la trazabilidad; cuáles son sus ventajas y desventajas en metodologías tradicionales y después mapear éstas a las metodologías ágiles.

Tradicionalmente, la trazabilidad de los artefactos del producto software ha tenido origen en los requerimientos y se han instanciado de diferentes maneras: productos específicos de gestión de requerimientos, bases de datos, hojas de cálculo o procesadores de texto convencionales [27].

Utilizando la trazabilidad, puede seguirse el historial de una característica implementada hasta las personas o grupos que la solicitaron, permitiendo un rápido análisis en cada fase del proyecto para [28]:

- Determinar la visión original y permitir una discusión controlada de los cambios en el alcance.
- Determinar qué elementos se verán afectados cuando consideramos agregar un nuevo requerimiento o modificar uno ya existente
- Verificar que el requerimiento contempla todos lo que el interesado solicitó.
- Verificar que la aplicación no implementa funcionalidades no demandadas por el cliente.

2.4 Un enfoque de trazabilidad ágil:

Para lograr desarrollar y mantener una trazabilidad útil en metodologías ágiles, Appleton [27] sugiere algunas pautas a tener en cuenta, como se menciona a continuación:

- Minimizar los artefactos de trabajo que se incluirán en la trazabilidad. Generar todos los requerimientos del software al inicio del proyecto implica un gran esfuerzo tanto de especificación como de trazabilidad [27]. Precisamente una de las características de los métodos ágiles es que los requerimientos de software se definen progresivamente durante el ciclo de desarrollo [5], por lo que definir el producto completo al inicio del proyecto significa esfuerzo extra que debe invertirse para mantener la trazabilidad de los requerimientos cada vez que surge un cambio en la definición del producto.
- Minimizar las fuentes de almacenamiento de información. Facilitar el acceso de los artefactos de trabajo a los miembros del equipo [27].
- Estabecer el nivel de granularidad de trazabilidad que será útil para el proyecto. Determinar el detalle que se necesita: realizar trazabilidad ¿a requerimientos o casos de uso?, ¿ a líneas de código, métodos, clases o módulos? [27].
- Automatizar la trazabilidad lo más que se pueda [27]. Las herramientas cumplen un rol muy importante en la trazabilidad, ayudan a crear y mantener

información, relacionar artefactos de trabajo y navegar dichas relaciones [28].

Tambien [27] menciona recomendaciones que ayudan a la trazabilidad:

- Herramientas de control de versiones e integración de éstas con las herramientas de gestión de productos [27]: Si los artefactos de trabajo que se generan durante el ciclo de desarrollo de software (requerimientos, modelos, código, casos de prueba, etc.) son almacenados en un repositorio, y éste se encuentra integrado con la herramienta de gestión del producto, entonces es posible la trazabilidad de las actividades a los artefactos de trabajo relacionados.
- Implementación de técnicas como el Desarrollo Guiado por Pruebas o TDD (por sus siglas en inglés *Test Driven Development*) [22] y el Desarrollo Guiado por Comportamiento o BDD (por sus siglas en inglés *Behavior Driven Development*) [23]. Estas técnicas permiten particionar el trabajo en tareas y relacionarlas con los artefactos de trabajo necesarios para implementar un requerimiento. Es decir, una misma tarea va a relacionar a todos los artefactos de trabajo requeridos para poder implementar un requerimiento (requerimiento, diseño, código, casos de prueba, etc.).

Por otro lado Beck [31] menciona que existe también un enfoque de trazabilidad que mapea a las historias de usuario con las pruebas de aceptación. Los dueños del producto deberán especificar las pruebas de aceptación que determinarán si las historias de usuario han sido implementadas o no [20]. Tanto los miembros del equipo como el dueño del producto deben acordar que existen un conjunto de pruebas de aceptación que demuestran que las historias han sido implemantadas satisfactoriamente. La relación entre las historias de usuario y las pruebas de aceptación constituyen una forma de trazabilidad. Podría no ser necesario mantener trazabilidad entre historias de usuario y código fuente, ya que si las pruebas pasan, uno puede asumir que existe código que implementa dicho requerimiento. Si posteriormente la rutina es modificada, las pruebas dejarán de pasar y deberán ser corregidas [31].

3 Resultados y discusiones

Siguiendo con la definición de requerimientos de software de la IEEE [15] citada previamente, podemos concluir que las historias de usuario son requerimientos ya que expresan el problema que el sistema o producto software debe resolver.

Sin embargo, las historias de usuarios no cumplen con todas las características planteadas en IEEE [13] según [21]. Para poder verificar la calidad de una historia de usuario surge el modelo INVEST [32] que presenta una serie de atributos a tener en cuenta para lograr escribir buenos requerimientos en metodologías ágiles.

El modelo INVEST [32] (por sus siglas en inglés *Independent, Negotiable, Valuable, Estimable, Small, Testeable*) es la clave para pensar y escribir buenas historias de usuario. Las historias deben ser Independientes, Negociables, Valiosa, Estimables, Pequeñas y Testeables [32].

- Independiente: Esto significa que las historias de usuario deben poder implementarse, probarse y entregarse por sí mismas sin depender de otras funcionalidades [32]. La dependencia entre las historias de usuario hace que sea más difícil planificar, priorizar y estimar. A menudo, se pueden reducir las dependencias haciendo una combinación de historias, o partiéndolas de forma diferente.
- Negociable: A diferencia de los requerimientos tradicionales, las historias de usuario no son obligaciones contractuales [32], sino más bien el compromiso de establecer conversaciones con el dueño del producto para convenir los detalles de los requerimientos y así poder implementarlos, probarlos y validarlos. Este es el proceso de negociación mediante el cual el equipo de desarrollo reconoce las

necesidades del negocio, pero también aporta sus ideas en base a la colaboración y retroalimentación.

- Valiosa: El objetivo de los equipos Scrum es proveer valor a los clientes y usuarios con los recursos disponibles, en el tiempo disponible. Ese es el motivo que hace que ésta sea la característica más importante del modelo INVEST [32]. Los ítems del producto son priorizados en función del valor que cada historia proveerá a los clientes, usuarios y demás *stakeholder*[2] del producto. Una forma muy eficaz de generar historias valiosas es hacer que el cliente las escriba.
- Estimable: Los desarrolladores necesitan poder estimar una historia de usuario para que se pueda priorizar y planificar. Los problemas que pueden impedirle a los desarrolladores estimar una historia son: falta de conocimiento del dominio (en cuyo caso se necesita más negociación / conversación); o si la historia es muy grande (en cuyo caso se necesita descomponer la historia en historias más pequeñas).
- Pequeña: Una buena historia debe ser pequeña en esfuerzo, generalmente representando no más de 2-3 personas/semana de trabajo. Una historia que es más grande va a tener más errores asociados a las estimaciones y al alcance. Las historias de usuario deberían ser lo suficientemente pequeñas como para poder ser implementadas en una iteración.
- Verificable: No se desarrolla lo que no puede ser probado. Si no puede probarse, nunca va a saberse si se ha terminado. Si una historia no puede ser verificada probablemente sea muy compleja, o tenga muchas dependencias con otras historias. Para lograr que las historias de usuario hayan sido probadas antes de finalizar la iteración, algunos equipos de desarrollo emplean un enfoque que comienza creando los casos de prueba y luego continúa con la codificación, este enfoque es conocido como Desarrollo Guiado por Pruebas (TDD) [22].

Desde el punto de vista de la utilidad de la trazabilidad en metodologías ágiles existen diferentes opiniones. Por un lado [33] menciona que las matrices de trazabilidad deberían ser requeridas sólo cuando existe un riesgo de perder información corporativa de valor. Si este no fuera el caso, entonces la trazabilidad se encuentra disponible en diversas formas: conversaciones, historias de usuario, código fuente, casos de prueba automáticos, documentos de diseño, reuniones diarias de avance, correos electrónicos, etc.

Por otro lado [27] reconoce el valor y la importancia de establecer y mantener trazabilidad de los requerimientos, pero no acepta los modos tradicionales de hacerlo.
Por su parte [34] menciona que en treinta años de experiencia en la industria del software, en investigación y en el gobierno, nunca ha percibido el retorno de inversión de las matrices de trazabilidad.

El desconocimiento sobre la importancia de la trazabilidad se manifiesta en lo poco que se sistematiza en esta área. La cultura del desarrollo de sistemas exige cada vez más documentación adecuada, sin embargo el mantenimiento y seguimiento de esta documentación no se lleva de manera apropiada, la trazabilidad es una forma de cambiar esta cultura y minimizar los fracasos en los proyectos de desarrollo de software [35].

Según [36] [21] las razones para hacer trazabilidad en proyectos ágiles se mencionan a continuación:

1. Porque existe información de las historias de usuario que no se documenta en el código: la razón u objetivo que se persigue con dicho requerimiento.
2. El valor que proporciona la funcionalidad.

[2] Un stakeholder es una persona que afecta o es afectada por el proyecto [44].

3. Porque los desarrolladores rara vez podrán recordar el propósito de las funcionalidades.
4. Porque es esencial establecer y mejorar el conocimiento del producto y de su dominio en la organización.

A pesar de que las metodologías ágiles fueron diseñadas para utilizar historias de usuario, el desarrollo de las aplicaciones está dirigido en su mayoría por requerimientos tradicionales [37]. Muchas de las organizaciones que subcontratan sus productos utilizan requerimientos como medio para definir y acordar lo que se debe construir. En estos casos, los equipos de desarrollo han tenido que encontrar el modo de trabajar con ambos a la vez. Las historias de usuario proveen flexibilidad en el diseño y en la implementación. Los requerimientos proveen precisión y facilidad de gestión [37]. Surge de este modo la siguiente pregunta ¿pueden coexistir ambos?. Los equipos deben estructurar el proyecto con el objetivo de aprovechar al máximo las ventajas de cada uno. Una forma de hacerlo es utilizando los requerimientos como el acuerdo entre los clientes y/o usuarios y el equipo de desarrollo, ya que los requerimientos son estructurados, objetivos y pueden definir exactamente lo que se debe implementar; y las historias de usuario, ya que pueden construir el contexto que ayuda a los equipos de desarrollo a entender cómo se va a utilizar la aplicación. Las historias de usuario pueden administrarse en forma conjunta con los requerimientos sin necesidad de invertir demasiado esfuerzo adicional si ambos representan las mismas funcionalidades desde diferentes puntos de vista; el del uso: las historias de usuario y el sistémico: los requerimientos [38]. Esto significa que, una historia de usuario podría estar relacionada con múltiples requerimientos y que un requerimiento podría a su vez, ser explicado por múltiples historias de usuario [37]. Los programadores utilizan las historias de usuario para diseñar e implementar las funcionalidades, y los requerimientos para agregar los detalles que sean necesarios. Los *testers*[3] escriben los casos de prueba que verifican los requerimientos, porque éstos últimos representan el acuerdo con el cliente, sin embargo, vuelven a las historias de usuario para comprender mejor cómo estructurar los casos de prueba y de este modo asegurarse que las funcionalidades más importantes han sido probadas en profundidad. Para que esto sea posible se deben relacionar de algún modo las historias de usuario y los requerimientos (trazabilidad) y asegurarse de que todos los requerimientos han sido mapeados a una o más historias de usuario. Se parte de un documento de requerimientos del producto que luego es descompuesto en funcionalidades, las cuales conocemos como historias de usuario [39].

4 Conclusión

Para lograr entender la importancia de la trazabilidad tanto en metodologías tradicionales como en metodologías ágiles es necesario comprender qué es un requerimiento, qué diferencia existe con la especificación de requerimientos y cuál es la necesidad de que los mismos sean expresados con cierto nivel de calidad. El requerimiento expresa el problema que el sistema o producto software debe resolver [21]. La especificación del requerimiento en cambio es el registro del requerimiento en una o más formas. Con el objetivo de definir los atributos y características deseables para la especificación de los requerimientos en metodologías ágiles, se revisaron los objetivos y las buenas prácticas de desarrollo de requerimientos según el estándar IEEE830 [6]. Inicialmente se había planteado la hipótesis de que las historias de usuario podrían ser compatibles con los atributos del estándar de la IEEE [6], sin embargo se encontró bibliografía que dice exactamente lo opuesto [21].

[3] Un tester es aquel que se dedica a realizar pruebas a nuevas aplicaciones o a modificaciones de aplicaciones existentes.

Durante el desarrollo del trabajo también se ha visto que las historias de usuario constituyen un enfoque de requerimientos que hace énfasis en la comunicación y participación de los usuarios en el proceso de desarrollo [21]. Los clientes rara vez conocen sus propias necesidades y durante el transcurso del proyecto esas necesidades y prioridades pueden cambiar. Las historias de usuario favorecen el trabajo en iteraciones lo cual facilita la adaptación a los cambios. También se ha visto que expresar los requerimientos de software como historias de usuario puede ocasionar algunos problemas. El arte de escribir buenas historias de usuario resulta muy difícil para quienes recién comienzan a emplear metodologías ágiles, los errores cometidos en esta etapa pueden llevar a desarrollar casos de prueba equivocados, a mal interpretar los requerimientos o peor aún a desarrollar el producto incorrecto [40]. Es por ello que se han creado modelos como el INVEST [32] que ayudan a escribir buenas historias de usuario definiendo atributos que las mismas deben cumplir.

Aún cuando las historias de usuario pueden pensarse como un reemplazo de los documentos de requerimientos tradicionales, es preciso recordar que la parte escrita de las historias de usuario permanece incompleta hasta tanto se efectúen las discusiones con los dueños del producto. Durante el desarrollo de este trabajo se ha visto que hay autores como Mike Cohn [21] que opinan que las historias de usuario no son obligaciones contractuales, y que por lo tanto no es necesario escribir los detalles conversados con los dueños del producto durante las reuniones de planificación. Las especificaciones del producto residen en lo que se ha implementado al finalizar cada iteración, en el código funcionando y en los casos de prueba. Pero también se ha visto que en algunos casos las historias de usuario se utilizan como punteros a requerimientos, típicamente en diagramas de flujos, hojas de cálculo donde se especifique cómo calcular un valor o cualquier otro artefacto que consideren el dueño del producto o los miembros del equipo [19]. Incluso en algunos casos coexisten los documentos de especificación de requerimientos y las historias de usuario.

Por otro lado se ha mencionado la importancia de mantener trazabilidad de requerimientos, pautas y recomendaciones [27] para el desarrollo y mantenimiento de la misma tanto en metodologías tradicionales como en metodologías ágiles. Sin embargo existen algunos autores [40] [33] [34] [4] que no reconocen del todo el valor y la utilidad de mantener trazabilidad en metodologías ágiles aunque [36] [21] mencionan tres razones muy importantes para hacerlo.

También se mostró que para metodologías tradicionales la IEEE [6] plantea como un criterio o atributo de calidad la trazabilidad de los requerimientos. Sin embargo, con respecto a las historias de usuario se encontró documentación que dice éstas no son compatibles con el mencionado estándar [21]. A pesar de que surge el modelo INVEST [32] para ayudar a la especificación de historias de usuario de calidad, este modelo no plantea como característica que las historias de usuario deban cumplir con el criterio de trazabilidad. La comprobación empírica sobre el uso de la trazabilidad en entornos ágiles es parte del trabajo futuro que se planea completar como se menciona en la siguiente sección.

5 Futuros trabajos

Para continuar con el plan de tesis, los pasos a seguir son: 1) Investigación acerca de las características que poseen los proyectos a gran escala; 2) Recopilación bibliográfica e investigación de lecciones aprendidas en la industria sobre trazabilidad de requerimientos de software en entornos de desarrollo ágiles mediante la realización de una encuesta para validar la hipótesis de la importancia real o no de realizar esta

práctica en metodologías ágiles; 3) Elaboración de un modelo de trazabilidad de requerimientos que contemple los diferentes modos de definición de requerimientos en entornos de desarrollo ágiles, según la exploración llevada a cabo en la primera etapa; 4) Aplicación del modelo de trazabilidad de requerimientos definido a un caso real (proyecto piloto) y finalmente la 5) Validación de la implementación llevada a cabo.

Agradecimientos

Queremos agradecer en especial a Mariano Zibbechi, Carlos Bertoni y Álvaro Ruiz de Mendarozqueta por su aporte en la revisión y corrección del presente trabajo.

Referencias

[1] R. Corral, «Exprimiendo Scrum: Scrum y la gestión de requisitos,» Los pensamientos, peleas y descubrimientos de Rodrigo Corral con Scrum, C++, C#, ASP.Net, Team System, Sql Server, Sharepoint, la arquitectura, la gestión de proyectos y el desarrollo de software en general..., 12 November 2007. [En línea]. Available: http://geeks.ms/blogs/rcorral/archive/2007/11/12/exprimiendo-scrum-scrum-y-la-gesti-243-n-de-requisitos.aspx. [Último acceso: 6 Junio 2012].

[2] Product Arts, «Agile Requirements - So What's Different?,» 2009. [En línea]. Available: http://www.product-arts.com/articlelink/204-agile-requirements-so-whats-different. [Último acceso: 9 Junio 2012].

[3] M. Fritzsche y P. Keil, «Agile Methods and CMMI: Compatibility or Conflict?,» *e-Informatica Software Engineering Journal*, vol. I, nº 1, 2007.

[4] S. W. Ambler, «Agile Requirements Modeling,» Ambysoft, 2010. [En línea]. Available: http://www.agilemodeling.com/essays/agileRequirements.htm. [Último acceso: 19 Junio 2012].

[5] S. W. Ambler, «Agile Requirements at Scale,» 22 Diciembre 2009. [En línea]. Available: https://www.ibm.com/developerworks/mydeveloperworks/blogs/ambler/entry/agile_requirements_at_scale?lang=en. [Último acceso: 19 Junio 2012].

[6] ISO/IEC/IEEE 29148:2011, «IEEE Recommended Practice for Software Requirements Specifications,» [En línea]. [Último acceso: 6 Diciembre 2012].

[7] O. Gotel y A. Finkelstein, «An Analysis of the Requirements Traceability Problem,» Imperial College of Science, Technology & Medicine, London, 1994.

[8] Software Engineering Institute, CMMI for Development, Version 1.3, Massachusetts: CMMI Product Team, 2010.

[9] Software Engineering Institute, «Requirements Development,» de *CMMI for Development, Version 1.3*, Pennsylvania, CMMI Product Team, 2010, pp. 325-340.

[10] M. S. Tabares, A. F. Barrera y J. D. Arroyave, «Un método para la trazabilidad de requisitos en el proceso unificado de desarrollo,» *Revista EIA: Escuela de Ingeniería de Antioquia*, pp. 69-82, Diciembre 2007.

[11] B. Ramesh y M. Jarke, «Toward Reference Models for Requirements Traceability,» *IEEE Transactions on Software Engineering*, vol. 27, nº 1, pp. 58-93, 2001.

[12] M. P. Izaurralde, «Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias de Usuario,» Trabajo de especialidad, Febrero 2013. [En línea]. Available: http://www.institucional.frc.utn.edu.ar/sistemas/lidicalso/pub/file/Tesis/Anteproyecto_Requerimientos_en_Metodolog%C3%ADas_Agiles.pdf.

[13] «SCRUM Alliance,» [En línea]. Available: http://www.scrumalliance.org/.

[14] Standards Coordinating Committee of the Computer Society of the IEEE, «IEEE Standard Glossary of Software

Engineering Terminology,» IEEE, New York, 1990.

[15] J. W. Brackett, *Software Requirements,* Boston University, 1990.

[16] A. Tuffley, «CIT3190 Information Technology Project Course,» 2000.

[17] Software Engineering Institute, «Requirements Management,» de *CMMI for Development, Version 1.3*, Pennsylvania, CMMI Product Team, 2010, pp. 341-347.

[18] M. Cohn, «Mountain Goat Software,» Mountain Goat Software, [En línea]. Available: http://www.mountaingoatsoftware.com/topics/user-stories. [Último acceso: 1 Diciembre 2012].

[19] M. Cohn, «An Overview,» de *User Stories Applied for Agile Software Development*, Indiana, Addisson - Wesley, 2009, pp. 3-15.

[20] M. Cohn, User Stories Applied: For Agile Software Development, Indiana: Addison Wesley, 2009.

[21] S. W. Ambler, «Introduction to Test Driven Development (TDD),» Agile Data, [En línea]. Available: http://www.agiledata.org/essays/tdd.html. [Último acceso: 9 Junio 2012].

[22] K. Pugh, Lean-Agile Acceptance Test-Driven Development, Boston: Addison-Wesley, 2010.

[23] M. Cohn, «What is Scrum?,» 2009. [En línea]. Available: http://www.mountaingoatsoftware.com/topics/scrum. [Último acceso: 3 Febrero 2013].

[24] J. Moccia, «Agile Requirements Definition and Management (RDM),» 27 Enero 2012. [En línea]. Available: http://onespring.net/blog/agile-requirements-definition-and-management-rdm/. [Último acceso: 11 Febrero 2013].

[25] Craig, «Requirements Traceability: Project Leadership, Requirements Management and Product Design,» 15 Enero 2008. [En línea]. Available: http://www.betterprojects.net/2008/01/requirements-traceability.html. [Último acceso: 9 Junio 2012].

[26] J. Ibañez, «Gestión de requerimientos IV: trazabilidad,» [En línea]. Available: http://blogs.salleurl.edu/project-management/gestion-de-requerimientos-iv-trazabilidad/. [Último acceso: 21 Agosto 2012].

[27] B. Appleton, S. Berczuk y R. Cowham, «Lean-Agile Traceability: Strategies and Solutions,» 19 Septiembre 2007. [En línea]. Available: http://www.cmcrossroads.com/article/lean-agile-traceability-strategies-and-solutions. [Último acceso: 21 Julio 2013].

[28] S. Berczuk, B. Appleton y R. Cowham, «The trouble with tracing: Traceability Dissected,» cmcrossroads, 30 Noviembre 2005. [En línea]. Available: http://www.cmcrossroads.com/article/trouble-tracing-traceability-dissected. [Último acceso: 22 Julio 2013].

[29] D. North, «Behaviour-Driven Development,» 2 Enero 2009. [En línea]. Available: http://behaviour-driven.org/. [Último acceso: 9 6 2012].

[30] K. Beck y M. Fowler, Planning Extreme Programming, Addison-Wesley, 2000.

[31] D. Leffingwell y P. Behrens, «A User Story Primer,» 2009. [En línea]. Available: http://trailridgeconsulting.com/files/user-story-primer.pdf. [Último acceso: 24 Enero 2013].

[32] M. Bolton, «Requirement Traceability Matrix and Agile Testing,» 14 Marzo 2008. [En línea]. Available: http://tech.groups.yahoo.com/group/agile-testing/message/13320?threaded=1&p=7. [Último acceso: 21 Agosto 2012].

[33] V. Hazrati, «Traceability Matrix in an Agile Project,» 6 Junio 2008. [En línea]. Available: http://www.infoq.com/news/2008/06/agile-traceability-matrix. [Último acceso: 21 Agosto 2012].

[34] M. M. Sandoval Carvajal y M. A. García

Vargas, «La Trazabailidad en el proceso de requerimientos de software,» 2008. [En línea]. Available: http://www.iiis.org/CDs2008/CD2008CSC/CISCI2008/PapersPdf/C601UZ.pdf. [Último acceso: 21 Agosto 2012].

[35] B. Rey-Mermet, «evocean,» 7 Enero 2011. [En línea]. Available: http://evocean.com/blog/tag/traceability/. [Último acceso: 21 Julio 2013].

[36] P. Varhol, «Conference Paper: Agility with Traceability: Blending Requirements and User Stories,» de *Quality Engineered Software & Testing Conference & Expo*, Chicago, 2012.

[37] C. Suscheck, «Defining Requirement Types: Traditional vs. Use Cases vs. User Stories,» 17 Enero 2012. [En línea]. Available: http://agile.techwell.com/articles/weekly/defining-requirement-types-traditional-vs-use-cases-vs-user-stories. [Último acceso: 20 Enero 2013].

[38] C. Langenfeld, «Using Rally to Map High Traceability User Stories: PRD to SRS,» 8 Julio 2011. [En línea]. Available: http://www.rallydev.com/community/agile-blog/using-rally-map-high-traceability-user-stories-prd-srs. [Último acceso: 20 Enero 2013].

[39] K. Kaczor, «5 Common Misktakes We Make Writing User Stories,» Scrum Alliance, 3 Agosto 2011. [En línea]. Available: http://www.scrumalliance.org/articles/366--common-mistakes-we-make-writing-user-stories. [Último acceso: 1 Diciembre 2012].

[40] B. Appleton, R. Cowham y S. Berczuk, «Lean Traceability: a smattering of strategies and solutions,» 18 September 2007. [En línea]. Available: http://www.cmcrossroads.com/article/lean-agile-traceability-strategies-and-solutions. [Último acceso: 21 Agosto 2012].

[41] K. Schwabe, Agile Project Management with Scrum, Microsoft Press, 2004.

[42] S. W. Ambler, «The Agile Scaling Model (ASM): Adapting Agile Methods for Complex Environments,» IBM Corporation, United States of America, 2009.

[43] UTN-FRC, «Universidad Tecnológica Nacional - Facultad Regional Córdoba,» [En línea]. Available: www.frc.utn.edu.ar.

[44] K. Wiegers., «Software Requirements,» 2nd Edition, Microsoft Press, 2003, 2003.

Datos de Contacto:
María Paula Izaurralde. Universidad Tecnológica Nacional. Facultad Regional Córdoba. Maestro M. Lopez esq. Cruz Roja Argentina Ciudad Universitaria - Córdoba Capital. paulaizaurralde@gmail.com

Natalia Valeria Andriano. Universidad Tecnológica Nacional. Facultad Regional Córdoba. Maestro M. Lopez esq. Cruz Roja Argentina Ciudad Universitaria - Córdoba Capital. nandriano@sistemas.frc.utn.edu