Trazabilidad de Procesos Scrum

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

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

Abstract. La trazabilidad es considerada en metodologías ágiles como un aspecto fundamental a estudiar para desarrollar sistemas de calidad. Sin embargo, los procesos ágiles ocurren en entornos donde no es frecuente encontrar un documento de especificación de requerimientos, no siendo posible aplicar técnicas clásicas de trazabilidad. En consecuencia, en este trabajo se propone un modelo de trazabilidad basado en las prácticas de Scrum. El objetivo principal del modelo es representar trazas existentes entre los artefactos generados durante procesos Scrum. La propuesta es especializada y ejemplificada siguiendo la documentación del proceso de desarrollo de Moodle.

Keywords: trazabilidad, scrum, ágil

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]. Los procesos ágiles ocurren en entornos donde no es frecuente encontrar un documento de especificación de requerimientos. Para capturar requerimientos muchas organizaciones utilizan Backlog Items, User Stories, Test, entre otras herramientas. Este escenario no permite aplicar técnicas clásicas de trazabilidad [4] como se venía aplicando en métodos pesados. Por lo tanto, en este caso surge el reto de compatibilizar un proceso ágil con una técnica rigurosa de métodos pesados, como lo es la trazabilidad, sin sacrificar agilidad. Para ello, es fundamental el desarrollo de modelos de referencia que permitan definir las adquisiciones y representaciones de trazas bajo la perspectiva ágil.

Los métodos ágiles [5], [6] 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. Estos métodos promueven la evolución y adaptación del software pero no están lo suficientemente enfocados en cómo pueden afectar los cambios en el proceso [7]. En la actualidad, una de las metodologías de desarrollo de software ágil más utilizada es

Scrum [8][9]. En consecuencia, en este trabajo se propone un modelo de referencia basado en las prácticas de Scrum. Teniendo como objetivo principal representar trazas existentes entre los artefactos generados durante el proceso Scrum. Esta representación facilitaría determinar si los requerimientos fueron cumplidos y además ayuda a evaluar el impacto de los cambios, entre otros beneficios [10].

A continuación se presenta una revisión de trabajos relacionados a la propuesta. Luego, en la sección 3 se representa el modelo propuesto, estructurado en un conjunto de vistas dadas por los constructores esenciales de Scrum. Esta representación permitirá responder diversas preguntas de competencias, tales como: ¿Qué artefactos son origen y fuente de los enlaces de trazabilidad?, ¿qué información de trazabilidad es representada por esos artefactos?, ¿en qué eventos es adquirida o modificada esta traza?, ¿cuáles stakeholders están involucrados en la creación, uso y mantenimiento de un artefacto? Las respuestas a estas preguntas facilitarán las tareas de trazabilidad que influyen directamente en el análisis del impacto de cambios, conformidad en el producto, obediencia del proceso, responsabilidad del proyecto, reproducibilidad de la línea base y aprendizaje organizacional [2].

Finalmente, en la sección 4 se detalla un caso de estudio basado en el proceso de desarrollo de Moodle [11] especializando el modelo propuesto, permitiendo refinar y validar la propuesta. Moodle es una plataforma de aprendizaje la cual utiliza Scrum como proceso de desarrollo.

2 Trabajos Relacionados

Como punto de partida para la propuesta del modelo de trazabilidad se instancia el metamodelo de trazabilidad, ilustrado en la Fig. 1 [4]. El metamodelo representa distintas dimensiones de la información de trazabilidad: información de trazas representadas, ¿dónde la información de trazabilidad es representada?, ¿cuándo esta información fue capturada o modificada?, stakeholders involucrados en la creación, uso o modificación de un objeto. Estas dimensiones son representadas por tres entidades principales *Source*, *Stakeholder* y *Object*.

Cada entidad y enlace en el metamodelo puede ser especializada e instanciada para crear modelos de trazabilidad específicos al proyecto u organización. Así mismo, nuestra propuesta instancia este metamodelo para representar la trazabilidad entre los artefactos generados, eventos, y roles involucrados en el proceso Scrum.

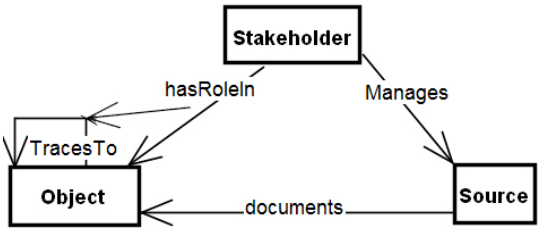

Fig. 1. Metamodelo de trazabilidad [4].

Es bien conocido que la documentación y el mantenimiento de la información de trazabilidad requiere de un gran esfuerzo de los desarrolladores. Los estudios existentes realizados sobre trazabilidad de software tradicional, están enfocados en reducir costos asociados con este esfuerzo manual, desarrollando ayudas automáticas en el establecimiento y mantenimiento de enlaces de trazabilidad entre artefactos de software [12], [13], [14]. La trazabilidad automática aplica técnicas de recuperación de información para generar enlaces candidatos. De esta forma se busca reducir considerablemente el efecto de los enfoques manuales, para construir y mantener una matriz de requerimientos de traza, y de igual forma proporcionar trazabilidad en los documentos legados [15], [16]. Estas propuestas también reflejan que implementando trazabilidad automática, el trabajo de los desarrolladores no debería verse incrementado de manera significativa. En este trabajo, se propone un enfoque diferente, se introduce un modelo inicial de trazabilidad de Scrum que integra los diferentes artefactos de Scrum (instanciados desde Object y Source, en la Fig. 1) e identifica los eventos de Scrum como instancias de la relación TraceTo (Fig. 1). A partir del modelo de Scrum propuesto podrían implementarse distintas herramientas de recuperación, evaluación y validación de trazas.

Otra contribución relevante en el área de trazabilidad es la propuesta de Cleland-Huang y colab. [17], donde se presenta un modelo genérico del proceso de trazabilidad y las futuras direcciones de la temática. Así mismo el modelo describe un conjunto de actividades claves de la trazabilidad como lo son planificación y gestión de la estrategia, creación, mantenimiento, y uso.

En la Fig. 2 se muestra un diagrama de estado con el modelo del proceso de trazabilidad, en el cual, son resaltadas aquellas actividades de trazabilidad que serán tratadas en nuestro modelo de Scrum. El proceso puede comenzar de dos maneras, cuando una traza es prevista o cuando es requerida. Este proceso continúa con la creación de trazas (*Creating*), en este estado se representan las tareas de: adquisición (*Acquiring Trace*), representación (*Representing Trace*), el posterior almacenamiento de las trazas (*Storing Trace*) y la validación de las mismas (*Validating Trace*). Una vez creada se puede continuar con el mantenimiento (*Maintaining*) o con el uso de esas trazas (*Using*). Luego proseguir con la actividad de evaluación, por último, el proceso finaliza cuando una traza es retirada o cuando el proyecto es archivado. Sin duda, para que este modelo sea viable en un ambiente ágil, como lo es Scrum, algunas actividades deben ser automatizadas, como la recuperación y almacenamiento de las trazas. A fines prácticos, el mantenimiento y la planificación de forma manual o semiautomática no podrían ser tenidas en cuenta, debido a que el trabajo del equipo de desarrollo se vería incrementado. Este modelo hace explícitas ciertas contradicciones entre los principios ágiles [5] y la trazabilidad, denotando la necesidad de un modelo que permita recuperar la trazabilidad sin sobrecarga en el proceso de desarrollo.

Las necesidades en los proyectos ágiles varían de uno a otro y por lo tanto sus restricciones de trazabilidad difieren de forma significativa. Espinoza y Garbajosa [2] remarcan este hecho proponiendo un enfoque basado en modelos para la planificación de la trazabilidad en los proyectos ágiles. Estos autores abogan por realizar trazabilidad ajustada a los objetivos, permitiendo al usuario establecer enlaces de trazabilidad. Los tipos básicos de trazabilidad son: trazabilidad hacia adelante y hacia atrás, tam-

bién vertical y horizontal, y las combinaciones de ambos pares. En la industria existen diferentes herramientas capaces de gestionar proyectos ágiles, integrar diferentes versiones de código, seguir requerimientos, etc. Según la clasificación de tipos de trazabilidad [18] estas herramientas implementan algunas técnicas como hipervínculos o tags para seguir un requerimiento de forma horizontal, hacia adelante o atrás y con una especificación de post requerimientos, de forma manual o semiautomática. En este trabajo se utilizarán algunas de estas combinaciones para poder responder a las preguntas de competencias planteadas en la sección anterior.

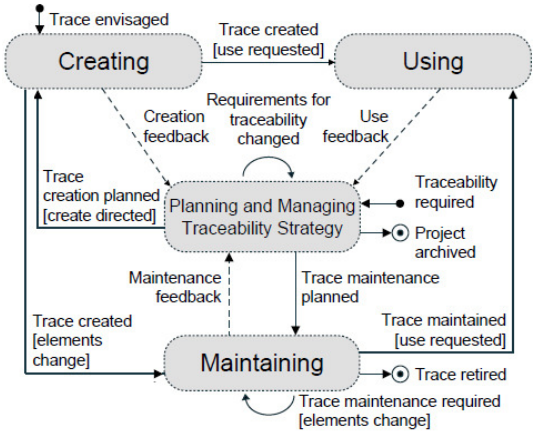

Fig. 2. Modelo del proceso de trazabilidad [17].

3 Modelo de trazabilidad de procesos Scrum

En esta sección, se propone un modelo de referencia de trazabilidad de Scrum. El modelo representa los constructores esenciales de Scrum: *roles*, *eventos*, *artefactos* y las reglas que permiten asociar estos conceptos. Estos conceptos son instanciados desde los conceptos de *Object*, *Source*, y la relación *TraceTo* del metamodelo de trazabilidad (Fig. 1) propuesto por Ramesh y Jarke [4].

Las reglas de Scrum asocian a los *eventos*, *roles*, y *artefactos* que gobiernan las relaciones entre ellos. Esto posibilita la obtención de vistas simplificadas del modelo, tales como: roles con artefactos, artefactos con eventos, y eventos con roles. Estos conceptos y sus relaciones nos permiten describir, de manera abstracta, las relaciones de trazabilidad implícitas en el proceso.

Los *roles* describen responsabilidades y niveles de autoridad de individuos, o grupos de individuos, que participan durante el proceso. En la Fig. 3 se representan los roles definidos en Scrum: *ScrumTeam*, *ProductOwner*, *ScrumMaster*, y *DevelopmentTeam*. 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.

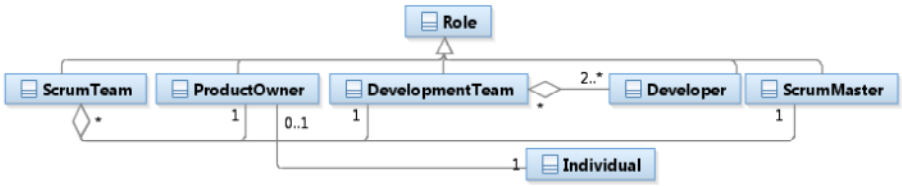

Fig. 3. Roles definidos en Scrum.

Los *eventos* 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. Son importantes en el modelado debido a que es en ellos donde la trazabilidad tiene lugar en el espacio temporal. En la Fig. 4 se representa a los eventos y las relaciones entre sí mediante una asociación *predecessor – successor*. A su vez toda reunión o ceremonia ocurre en un determinado intervalo de tiempo, *TimeBox*. Los eventos son definidos como instancias de la relación *TraceTo* (Fig. 1), ellos enlazan dos instancias del concepto *Object* (Fig. 1), conformando la tripleta de elementos de traza: (*artefacto fuente*; *evento* enlace de *traza*; *artefacto destino*).

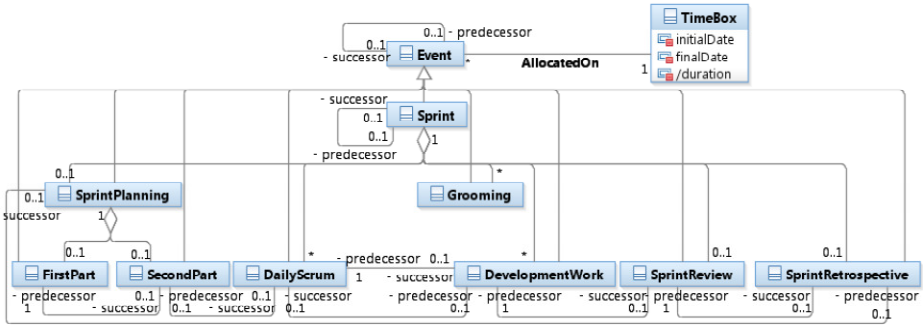

Fig. 4. Eventos definidos en Scrum.

Por definición un *artefacto* es la definición de un producto de trabajo [12]. Los artefactos pueden estar compuestos por otros artefactos y son productos de trabajo concretos consumidos, producidos o modificados por los distintos *eventos* del proceso Scrum. Así de esta manera, podemos referirnos a la actividad de creación de traza (*Creating*, Fig. 2) para definir un artefacto fuente, un artefacto destino, y un enlace entre ellos. Algunos de estos enlaces pueden estar reificados y ser modelados como artefactos. Por ejemplo, en la Fig. 5 se define una tripleta de trazabilidad entre *ProductBacklogItem* (*fuente*), *ProductIncrement* (*destino*), y *SprintBacklog* (*TraceTo*). *SprintBacklog* reifica el enlace entre fuente y destino, permitiendo trazar qué *ítem* fue considerado en un *incremento* realizando un *Sprint* particular. Es posible definir una regla de derivación en OCL para inferir esta relación. En la ecuación (1) se especifica dicha regla.

context ProductBacklogItem:target: Set(ProductIncrement)

derive: self.sprintBacklog.productIncrement (1)

Además, ciertas relaciones entre artefactos permitirían inferir nuevas relaciones de trazas. Por ejemplo, la relación *RefineOn* entre *ProductBacklogItems* y la relación existente entre *ProductBacklogItem* y *Task*. Si un ítem es considerado en un Sprint para producir un incremento de producto, entonces, el ítem padre de este deberá también ser trazado con ese incremento, este hecho puede ser reflejado mediante la ecuación (2).

context ProductBacklogItem:targetInf:Set(ProductIncrement)

derive: self.target->union(self.subItem.targetInf) (2)

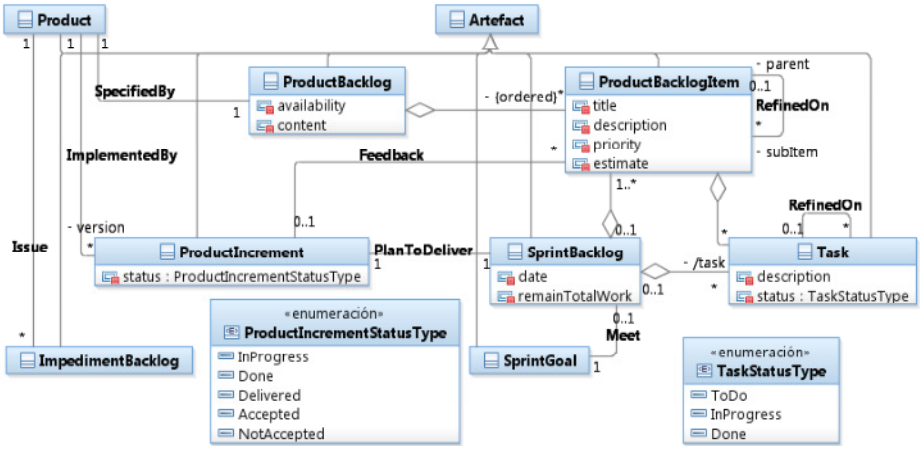

Fig. 5. Artefactos definidos en Scrum.

3.1 Relaciones entre Roles, Eventos y Artefactos

Las relaciones entre los distintos constructores de Scrum (*roles*, *artefactos* y *eventos*) permiten a los Stakeholders (Fig. 1) responder algunas preguntas sobre cómo un proceso Scrum fue llevado a cabo. Por ejemplo, ¿Cuál Stakeholder fue responsable de un artefacto? En Scrum, un rol es responsable de un artefacto; esto simplifica trazar a los distintos responsables de los artefactos. Sin embargo, otros roles pueden usar y actualizar, artefactos con el permiso adecuado. La Fig. 6(a) explicita las relaciones existentes entre los roles y artefactos.

En la Fig. 6(b) se ilustra la relación entre eventos y artefactos. Un evento culmina con la entrega de un artefacto, generalmente representado como asociación *Output* en Fig. 6(b). De esta manera, los conceptos de eventos reifican la relación *TraceTo* mostrada en la Fig. 1. Por ejemplo, el evento *Sprint* permite trazar objetos de entrada, como *ProductBacklogItem*, en objetos de salida, tales como *ProductIncrement*. Esta

traza nos permitiría conocer qué requerimientos fueron considerados en un incremento de producto en particular.

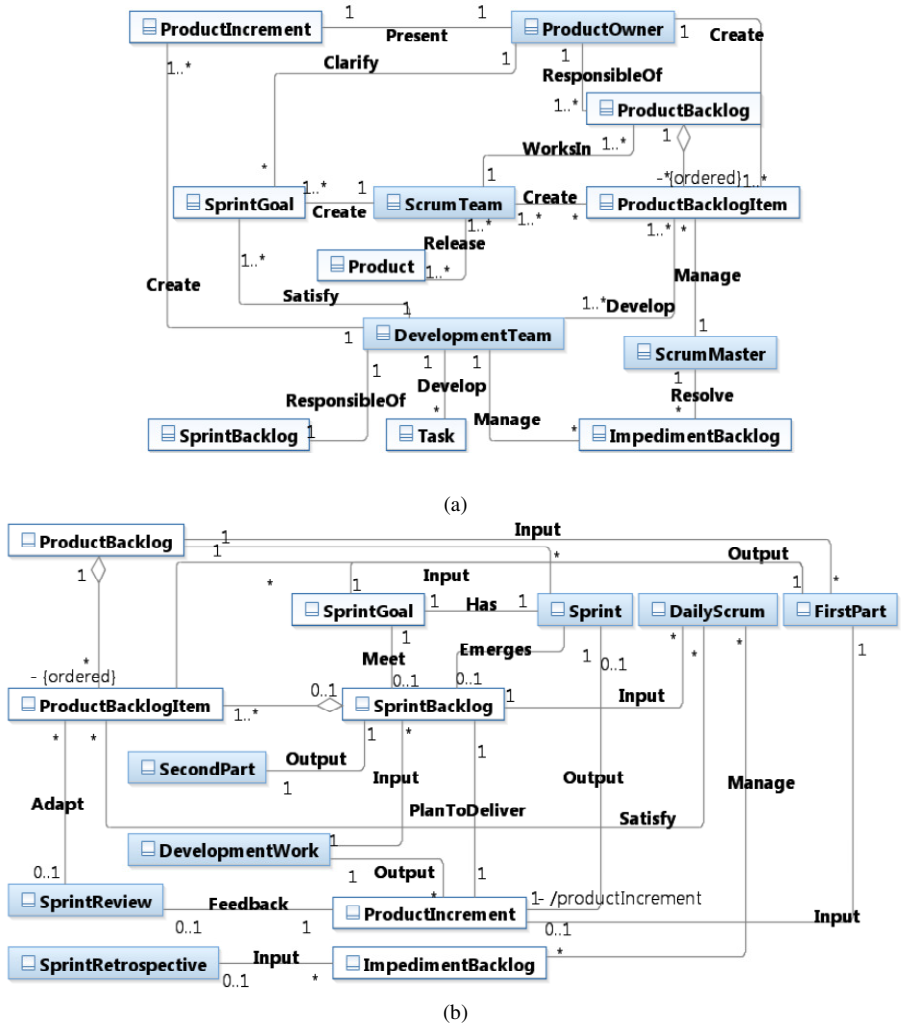

Fig. 6. Representación de Scrum. (a) Roles y artefactos. (b) Eventos y Artefactos

En ambos diagramas de la Fig. 6 se incluyen diversas relaciones entre roles, artefactos y eventos. Se visualizan los distintos roles encargados de administrar los artefactos generados durante procesos Scrum. Estos artefactos a su vez participan en potenciales trazas (artefactos tipo *Object* y relaciones *TraceTo* en Fig. 1). Cabe destacar que el *DevelopmentTeam* debería ser el responsable de la planificación de la trazabilidad. Sería necesario definir las necesidades de las trazas, qué relaciones definidas en los modelos introducidos realmente se van a capturar, usar y mantener durante el proceso de desarrollo.

4 Especialización del modelo propuesto en un caso real

Para validar nuestra propuesta se analiza el proceso de desarrollo de la plataforma de aprendizaje Moodle [11]. Esta plataforma está diseñada con el objetivo de brindar un sistema integrado que permita crear ambientes de aprendizaje personalizados. Este es un proyecto dirigido por colaboración abierta, utilizando Scrum como proceso de desarrollo.

En Moodle los Stakeholders interpretan diferentes roles tales como: *MoodleTeam*, *MDeveloper*, *MScrumMaster*, *MProductOwner*, *IntegratorTeam*, *Tester*, *Product-Mantainer* y *ComponentLead*. En la Fig. 7 se representan esos roles especializando los conceptos de roles de Scrum (Fig. 3). A su vez se encontrarían involucrados en el proceso de trazabilidad (Fig. 2) que abarca desde la determinación de necesidades, pasando por planificación e implementación, finalizando con la evaluación y retrospectiva del proceso. El *MoodleTeam* (un *ScrumTeam*, Fig. 7) es el responsable de la creación y mantenimiento de los *Issues* (introducido en Fig. 8), que una vez desarrollados serán parte de un *ReleaseCandidate*, pudiendo dar origen a nuevos *Issues* o *Bugs*. Esto refleja que prácticas de trazabilidad son llevadas a cabo de manera implícita y la recuperación, análisis, actualización y almacenamiento de estas trazas debe ser realizada de forma completamente automática, sino estarían generando un trabajo adicional en los desarrolladores.

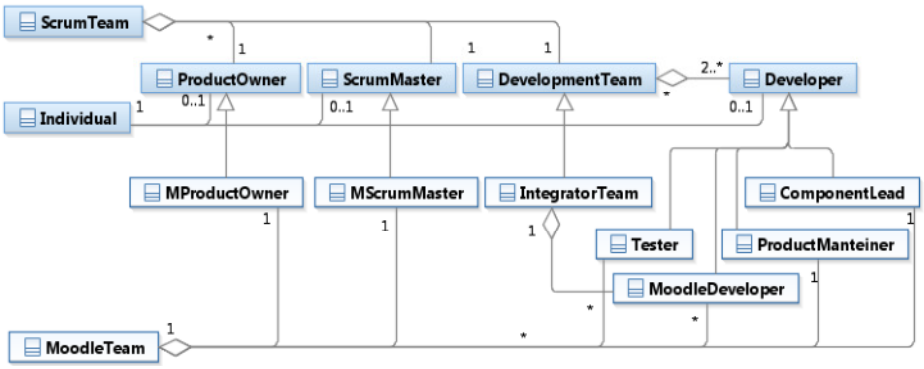

Fig. 7. Roles de Moodle en Scrum.

Los artefactos de Moodle creados durante su proceso de desarrollo son: *Issues*, *List of Issue*, *Issues on Sprint*, *Release Candidate*, *Release*, *Roadmap*, y *Bug Issue* (Fig. 8). La información representada por estos artefactos muestra que distintos usos básicos de trazabilidad pueden implementarse en el modelo [18]. Un trazado hacia adelante o hacia atrás siguiendo, o no, un camino cronológico del proceso de Scrum (relaciones *predecessor-successor*), o un trazado hacia atrás siguiendo un camino inverso del proceso (relación *successor-predecessor*). Esto permitiría descubrir en qué *eventos* fueron creadas, modificadas o eliminadas las *trazas*. Por ejemplo un requerimiento representado por un *Issue* (Fig. 8), creado en el evento *FirstPart*, puede ser luego refinado en nuevos *Issues* en el evento de *Grooming* (Fig. 8). Posteriormente puede

ser pronosticado para ser desarrollado en un *Sprint*, finalmente implementado y aprobado en un *ReleaseCandidate*. Este modelo nos ayuda a encontrar las relaciones existentes entre *Issues* (*ProductBacklogItems*), *Release Candidates* (*ProductIncrements*), etc., permitiéndonos especificar la granularidad de la información necesaria para las trazas. Esta información puede ser creada durante *Issue Triage* (especializando el evento *Grooming* de Scrum) mientras *Issue* y *Task* están siendo analizados.

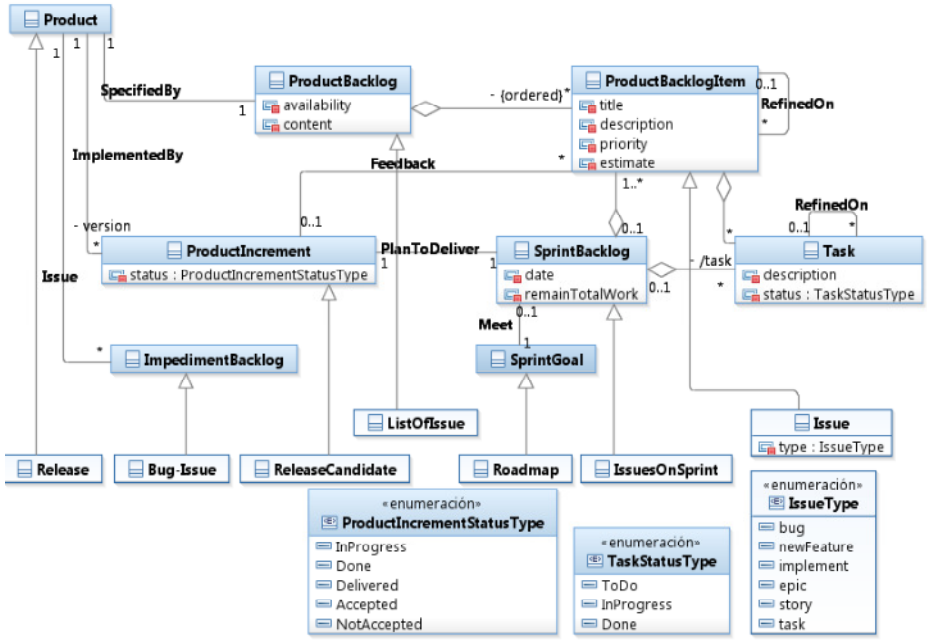

Fig. 8. Artefactos de Moodle en Scrum.

Moodle utiliza el sistema de seguimiento de proyectos Issues Atlassian Jira. Esta herramienta de gestión utiliza el The Moodle Tracker [19], que es una base de datos encargada de almacenar y gestionar todos los issues, errores, mejoras y solicitudes de características correspondientes al desarrollo del proyecto y sus sistemas relacionados. Tracker se conecta a diferentes bases de datos de subproyectos tales como: Moodle, Moodle Community Sites, Moodle Mobile, Moodle QA, Moodle Testing, y Plugins.

La base de datos del proyecto Moodle tiene 47.747 tuplas (*Issues*). Cada issue está definido mediante 28 atributos, esto resultaría en 1.336.916 de datos si cada campo estuviera completo. Existen 7 atributos con campos completamente vacíos, además hay 13 tuplas corruptas y de los restantes atributos no todos poseen sus campos completos. Por lo tanto, se filtró la base de datos con valores no nulos, resultando 790.797 elementos. En particular, en el presente trabajo se muestra información consultada desde la base de datos Mobile, en la que existen 906 tuplas con 19 propiedades no nulas, resultando en un total de 15.105 elementos.

La Fig. 9 muestra una captura de pantalla de Jira Client con una conexión a Moodle Tracker – Mobile. Realizando consultas a la base de datos de Mobile podemos responder a las preguntas de competencia que guían este trabajo. Tales consultas indican que el *ListOfIssue* contiene 906 ítems, además existen 27 *ReleaseCandidate* a lo largo del proyecto, y en el *Release versión 2.0* apreciamos que 43 *IssuesOnSprint* están afectados al *Sprint* que la generó. A su vez se observan aquellos *MoodleDeveloper*, *MComponetLead*, entre otros stakeholders que tenían asignado al menos un *Issue*, y aquellos que participaron en desarrollo de alguno. De los 43 *Issue/Task* solo 7 tienen *status Done* (3 cerrados y 4 bugs resueltos), 12 tienen *status InProgress* y hay 24 *Issues ToDo*. Para aquellos 24 *ToDo* se realizó un nuevo filtro revelando a qué componente pertenecen, a qué tipo de tarea corresponden y su cantidad. En la parte inferior izquierda de la ventana de la Fig. 9 se presenta una tabla indicando los 24 *Issues* abiertos en los componentes *Android*, *Ionic*, *Settings*, y *Others*. La primer columna representa *New features*, la segunda *task*, y la tercera *sub-tasks*.

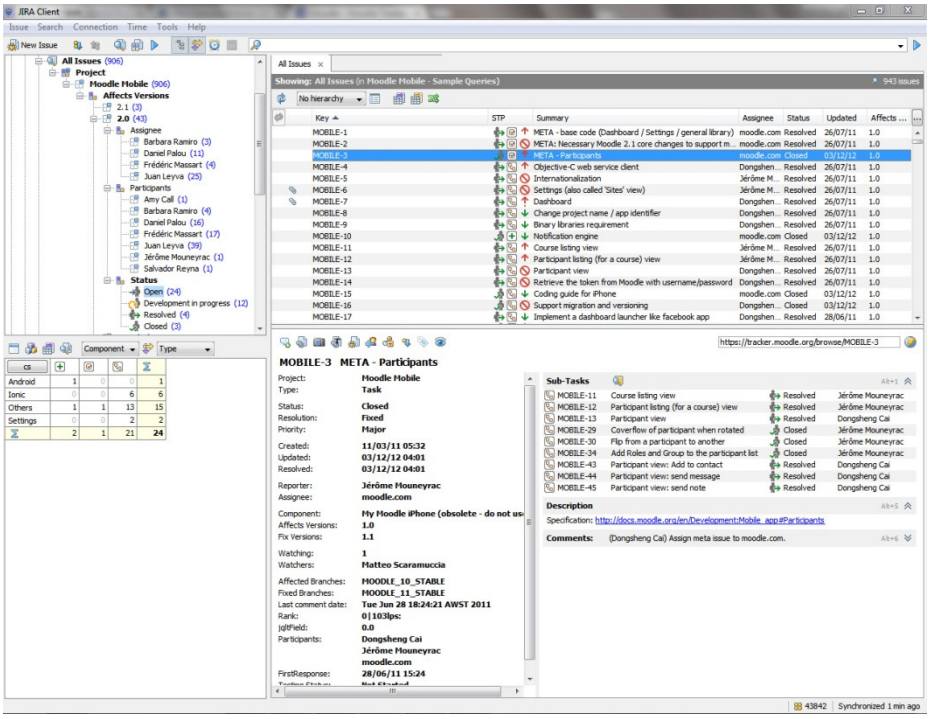

Fig. 9. Issues en Moodle Mobile.

En la parte derecha de la ventana de la Fig. 9 se visualizan todos los *Issues* del proyecto Moodle Mobile y en particular, en la parte inferior se visualiza un *Issue* concreto, el *Issue MOBILE-3*, de tipo *Task*, y sus descendientes (*sub-tasks*). El *Issue MOBILE-3* es definido a partir de la versión *1.0* y planificado para ser desarrollado en un *Sprint* afectando a la versión *1.1* del *Release*. Además el mismo fue refinado (*RefineOn*) en nuevas *Sub-Tasks*, como los son *MOBILE-11, 12, 13, 29, 30, 34, 43, 44* y

45. Otra información apreciable es cuáles Stakeholders participaron en su desarrollo. El *Issue* pudo ser marcado con *Status* "*Closed*" debido a que todas las tareas que lo componen fueron marcadas como "*Closed*" o "*Resolved*".

Para responder a las preguntas introducidas en la sección 1, es trascendental mencionar que un *Issue* es definido como una especialización de *ProductBacklogItem* (Fig. 8). A partir de tal información, podemos conocer las versiones que fueron afectadas por un Issue en particular (target en ecuación 1, targetInf en ecuación 2) o qué requerimientos (*ProductBacklogItems*) guiaron a los diferentes *ReleaseCandidates* (*ProductIncrements*).

Los *Issues* son adquiridos durante la realización de diversos eventos de Moodle. Por razones de espacio no se incluye la especialización de eventos según Moodle, no obstante se indica que los eventos que permiten generar los *Issues* son: *PlanningAndEstimation* (especialización de *SprintPlanning*), *IssueTriage* (especialización de *Grooming*), *PlanningDevelopment* (especialización de *DailyScrum*), *HQDailyScrum* (especialización de *DailyScrum*), y *ReleaseCandidateEvaluation* (especialización de *SprintReview*). Las trazas luego son modificadas en *IssueTriage*, *HQDailyScrum*, *BugFixing* (especializaciones de *DevelopmentWork*), *QA* y *Testing* (especializaciones de *DevelopmentWork*).

Los stakeholders involucrados en el proceso de desarrollo de Moodle son los encargados de administrar los artefactos generados durante el proceso. Estos artefactos a su vez participan en potenciales trazas (artefactos tipo *Object* y relaciones *TraceTo* en Fig. 1). La recuperación y evaluación de los enlaces es realizada de forma manual o semiautomática. Existen herramientas que permiten encontrar relaciones entre artefactos definidas de manera explícita entre ellos. Si la trazabilidad fuera planificada de manera consciente, se podrían implementar plugins para herramientas como JIRA que faciliten la recuperación, representación y evaluación de los enlaces en forma automática.

Asimismo, existen relaciones de trazas que no están en forma explícita disponibles en la aplicación JIRA, pero mediante su planificación y definición sería posible especificar consultas que permitan recuperarlas automáticamente. Por ejemplo, Un *Issue*, que es un *ProductBacklogItem*, puede ser refinado en nuevos *Issues*. En la Fig. 9 se ilustra un ejemplo posible de esta relación, definiendo que un *Issue* de tipo *task* es descompuesto en un conjunto de *sub-tasks*. Tal como se ilustra en la figura, *Task MOBILE-3* afecta el componente *My-Moodle IPhone* y se espera su incorporación en la *Release 1.1*. Esta información es ilustrada en el diagrama de objetos incluido en la Fig. 10. En este diagrama, una de sus *sub-tasks*, *MOBILE-11*, prevé ser finalizada en la *Release 1.0*. En consecuencia, parcialmente el *Issue MOBILE-3* será resuelto en la *Release 1.0* y no sólo en la *Release 1.1*, como indica en forma explícita la herramienta (Fig. 9). Este tipo de información estaría representada por *targetInf* definida en la ecuación (2).

Existen otras relaciones que se presentan en otros *Issues* y pueden ser inferidas, por ejemplo, en el *Issue MOBILE-153*, del tipo *New Feature*, se prevé migrar de UMM (Unofficial Moodle Mobile) a MM (Moodle Mobile). En la Fig. 11 se incluye la información sobre este *Issue*, se planifica para la *Release 1.2*, afectará al componente *Others* (se usa este término cuando afecta a varios y no indica cuáles) y es descom-

puesto en 32 *Issues* de tipo *Task*. Estos 32 *Issues* se planifican en las *Release 1.2, 1.3, 1.4* y *1.8*. En consecuencia, el *Issue MOBILE-153* debería ser trazado a las *Release 1.2, 1.3*, *1.4,* y *1.8*. Además, los 32 *Issues* planean afectar diversos componentes, tales como *Android* y *iOS*. De este modo, el *Issue MOBILE-153* debería ser trazado a estos componentes. En la Fig. 12 se ilustra mediante un diagrama de objetos esta situación, dado el número de *Issues*, sólo se representa al *Issue MOBILE-153* y dos de sus *sub-Task*: *MOBILE-165* y *MOBILE-168*.

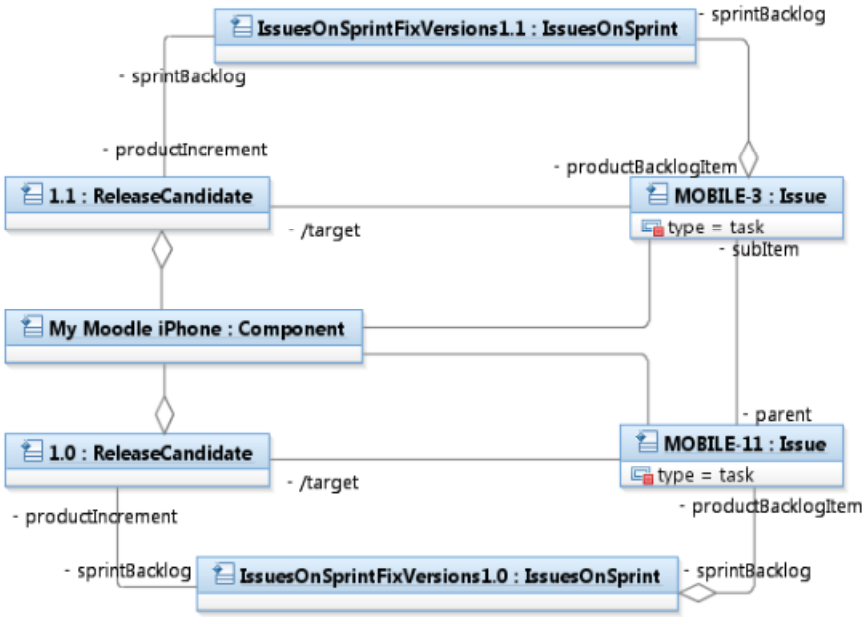

Fig. 10. Diagrama de objetos Issue MOBILE-3 y una sub-task, MOBILE11.

5 Conclusiones

En este trabajo se presentó un modelo de trazabilidad de Scrum. Este es definido como una instancia del metamodelo de trazabilidad (Fig. 1). Nuestra propuesta naturalmente integra la representación de los eventos (relación *TraceTo*) del proceso Scrum con los artefactos (concepto *Object*) que ellos generaron, capturando también la participación de los actores (*roles*). El modelo fue extendido para representar un proceso de Scrum particular. Como primer resultado, el modelo propuesto muestra que algunas prácticas de Scrum pueden ser interpretadas como prácticas de trazabilidad. Sin embargo, es necesario un mecanismo formal que nos permita automatizar la recuperación de enlaces de trazabilidad en el desarrollo ágil. Nuestra propuesta es un primer paso en esta dirección, pero estas ideas primarias necesitan ser mejoradas y aún más trabajadas en el futuro.

Asimismo este modelo refleja una serie de actividades que podrían servir para establecer la trazabilidad de artefactos y hacerla usable, como así también una descrip-

ción de los responsables de esas tareas y los recursos que serán necesarios para llevarlas a cabo. Existen herramientas simples como sistemas de control de versiones, wikis, herramientas de rastreo de cambios, entre otras que permiten el hacer uso de trazabilidad sin perder agilidad. El uso de estas herramientas no pretende sobreponerse a los principios ágiles porque buscan facilitar la tarea del desarrollo y de la satisfacción del cliente.

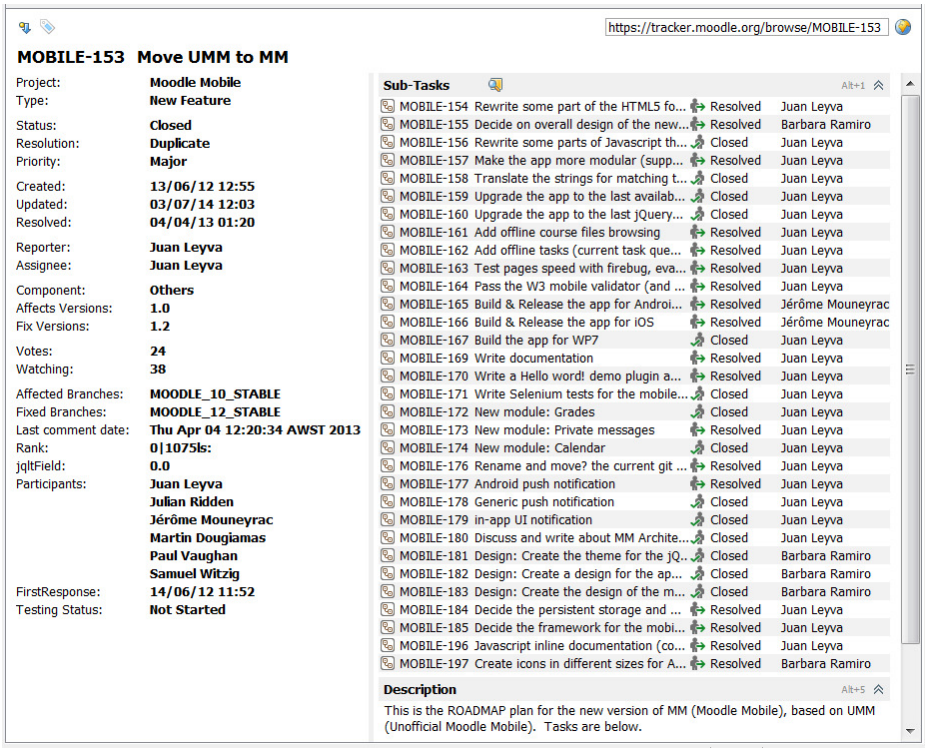

Fig. 11. Issue MOBILE-153.

Se puede decir que en Scrum las trazas son creadas entre los diferentes artefactos pero esto no implica que exista trazabilidad, debido a que las otras partes del proceso de trazabilidad no son cumplidas (uso, planeamiento de la estrategia y mantenimiento). Además de la creación de los enlaces es importante definir la recuperación de los enlaces y la planificación de la trazabilidad para que su uso sea realmente útil, y no tan solo un sobre-trabajo. Si en Moodle se quisiera aplicar técnicas de trazabilidad, un aspecto fundamental a estudiar es la calidad de esos enlaces que se pretende trazar.

En proyectos como Moodle se vuelve difícil, sino imposible, el poder confiar en la memoria (individual o colectiva) para entender cuáles funciones específicas fueron implementadas en código, o recuperar partes del código que son sensibles a cambios debido a los intercambios interrelacionados. Para solucionar esta dificultad se puede realizar una combinación de diferentes técnicas como trazas básicas desde user stories

a casos de pruebas y a código. Otra técnica es la trazabilidad justo a tiempo y la última podría ser user stories más refinadas a trazas de código con utilización de tags.

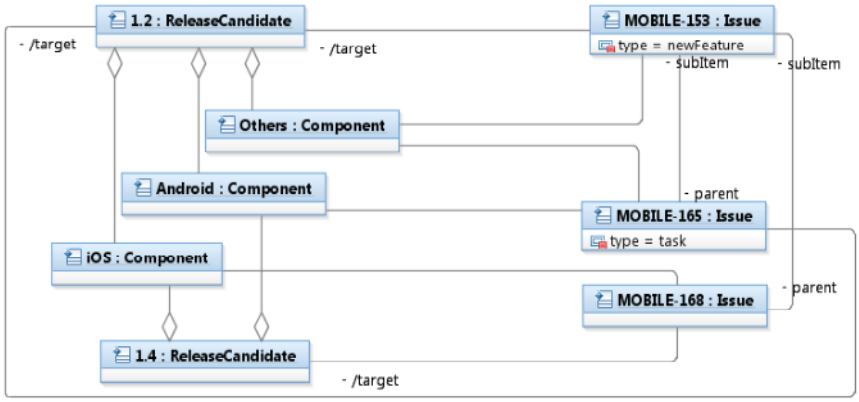

Fig. 12. Diagrama de objetos Issue MOBILE-153 (parcial).

6 Referencias

1. Cao, L., Ramesh, B.: Agile Requirements Engineering Practices: An Empirical Study. IEEE Softw. 25(1), 60–67 (2008)
2. Espinoza, A., Garbajosa, J.: A study to support agile methods more effectively through traceability. Innov. Syst. Softw. Eng. 7(1), 53–69 (2011)
3. ISO/IEC, IEEE: Systems and software engineering – Vocabulary. ISOIECIEEE 247652010E, (2010)
4. Ramesh, B. Jarke, M.: Toward reference models for requirements traceability. IEEE Trans. Softw. Eng. 27(1), 58–93 (2001)
5. Manifesto for Agile Software Development, http://agilemanifesto.org/ (2001)
6. Williams, L.: What Agile Teams Think of Agile Principles. Commun ACM 55(4), 71–76 (2012)
7. Ayed, H., Habra, N., Vanderose, B.: Agile processes evolution challenges. In: BENEVOL (2013)
8. Schwaber, K.: SCRUM Development Process. In: OOPSLA Business Object Design and Implementation Workshop, pp. 117–134 (1997)
9. Sutherland, J., Schwaber, K.: The Scrum Guide, http://www.scrumguides.org/ (2013)
10. Mäder, P. Gotel, O.: Ready-to-Use Traceability on Evolving Projects. In: Software and Systems Traceability, pp. 173–194. Springer (2012)
11. The Moodle Project. http://www.moodle.org (2014)
12. Arkley, P., Mason, P., Riddle, S.: Position paper: Enabling traceability. In: Proceedings of the 1st International Workshop on Traceability in Emerging Forms of Software Engineering, pp. 61–65 (2002)
13. Zhang, Y., Witte, R., Rilling, J., Haarslev, V.: Ontological approach for the semantic recovery of traceability links between software artefacts. IET Softw. 2(3), 185–203 (2008)

14. Goknil, A., Kurtev, I., van den Berg, K.: Generation and validation of traces between requirements and architecture based on formal trace semantics. J. Syst. Softw 88, 112-137 (2014)
15. Cleland-Huang, J., Settimi, R., Romanova, E., Berenbach, B., Clark, S.: Best Practices for Automated Traceability. Computer 40(6), 27–35 (2007)
16. Mahmoud, A.: Toward an effective automated tracing process. In: IEEE 20th International Conference on Program Comprehension (ICPC), pp. 269–272 (2012)
17. Cleland-Huang, J., Gotel, O. C. Z., Huffman Hayes, J., Mäder, P., Zisman, A.: Software Traceability: Trends and Future Directions. In: Proceedings of the on Future of Software Engineering, pp. 55–69 (2014)
18. Gotel, O., Cleland-Huang, J., Hayes, J. H., Zisman, A., Egyed, A., Grünbacher, P., Dekhtyar, A., Antoniol, G., Maletic, J., Mäder, P.: Traceability Fundamentals. In: Software and Systems Traceability, pp. 3–22. Springer (2012)
19. The Moodle Tracker. http://tracker.moodle.org/ (2015)

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