Recibido: 31 de agosto del 2018
Aceptado: 10 de octubre del 2018
doi: 10.26439/interfases2018.n011.2953

Automatización de requisitos: historias de usuario generadas a partir de un modelo orientado a objetivos basado en el *framework* i*

Delicia Esmeralda Lucero Guevara
delicia.lucero.guevara@gmail.com
Universidad Nacional Mayor de San Marcos, Lima, Perú

Nora Bertha La Serna Palomino
nlasernap@unmsm.edu.pe
Universidad Nacional Mayor de San Marcos, Lima, Perú

Resumen

Este artículo describe la implementación de una herramienta para la generación automatizada de historias de usuario a partir de un modelo gráfico. Se aplica el enfoque de modelado orientado a objetivos, el cual utiliza la notación i*. El fundamento principal es que el objetivo del negocio no es el que cambia, sino que lo hacen las maneras en que este se puede lograr. Una vez recopilados los requerimientos a través de este modelo se generan las historias de usuario de manera automática para la posterior trazabilidad. Estas cumplen con el estándar base propuesto en las metodologías ágiles, asegurando que al llegar a manos del desarrollador no existan ambigüedades en ninguna de las dos perspectivas: la gráfica y la narrativa. Los resultados de las pruebas con desarrolladores y analistas, así como la aplicación de un cuestionario, demostraron que la herramienta propuesta contribuye directamente al establecimiento de requisitos precisos, completos y consistentes, optimizando el empleo del tiempo de los miembros del equipo.

Palabras clave: ***requerimientos, historia de usuario, metodologías ágiles, modelamiento orientado a objetivos***

Abstract

Automation of requirements: user stories generated from an objective-oriented model based on the notation i*

This paper describes the implementation of a tool for automated generation of user stories from a graphical model. The objective-oriented modeling approach is applied, which uses the notation i*. The main rationale is that the business objective is not the one that changes, but the ways in which it can be achieved.

Once the requirements are compiled through this model, user stories are generated automatically for later traceability. These comply with the basic standard proposed in agile methodologies ensuring that, when arriving at the hands of the developer, there are no ambiguities in either of the two perspectives: graphical and narrative.

The results of the tests with developers and analysts, as well as the application of a questionnaire, showed that the proposed tool contributes directly to the establishment of precise, complete and consistent requirements, optimizing the time used by the team members.

Keywords: ***requirements, user history, agile methodologies, objective-oriented modeling***

1. Introducción

En el campo de la ingeniería de *software* la elicitación es una fase inicial del análisis de requerimientos que debe estar exenta de ambigüedades con el fin de alcanzar un producto de calidad. Analizar los requerimientos demanda un alto grado de interacción entre las partes involucradas para intercambiar información. El ingeniero de requisitos ayuda a los interesados a identificar los requerimientos y el entorno que satisface sus necesidades, que se expresan en un lenguaje natural y sencillo.

El proceso de análisis de requerimientos es de un nivel más alto que el proceso de diseño del sistema técnico en sí, ya que en el primero se evalúan las alternativas con respecto a los objetivos, mientras que en el segundo los objetivos se pueden usar para ayudar a generar sistemáticamente soluciones potenciales.

Actualmente el desarrollo de las aplicaciones *web* sigue metodologías que lo orientan hacia la creación de componentes a medida, la adquisición de componentes de terceros o ambos. Esto ocasiona que exista un número creciente de usuarios expuestos a diversas perspectivas referidas al mismo proceso de negocio, que van desde las interfaces hasta los propios conceptos utilizados en las aplicaciones. También puede provocar que los administradores de las aplicaciones tengan distintos componentes que ofrecen la misma funcionalidad y documentación diversa para definir el mismo proceso.

Aun cuando pareciera que el modelo gráfico y la historia de usuario tienen el mismo propósito, es cierto también que sus perspectivas tienen una representación y léxico propio de analistas versus diseñadores, y que en la interpretación de uno u otro se podría perder la objetividad, distorsionando la necesidad inicial del cliente.

Actualmente los escenarios en los que se desarrollan las aplicaciones son tan volátiles que existe apremio por incorporar nuevos requisitos, modificar requisitos existentes o eliminar algunos que dejan de ser necesarios. Esta situación propicia que se pierda la traza entre el modelo gráfico y la historia de usuario, puntualmente cuando los artefactos que recopilan la información se encuentran aislados.

En este artículo se presenta el marco teórico de la investigación; así como la metodología y técnica que se ha utilizado en el desarrollo de una herramienta gráfica para la automatización de generación de historias de usuarios desde un modelo orientado a objetivos basado en i*. Luego se describe la herramienta de soporte desarrollada y finalmente se presentan los resultados de las pruebas y encuesta aplicadas a los especialistas.

2. Marco teórico

2.1 Ingeniería de requisitos

La ingeniería de requisitos comprende dos procesos principales: a) desarrollo de requisitos, que involucra la captura, análisis, especificación y validación de los requisitos, y b) gestión de requisitos, que es un proceso referido a su trazabilidad (Bourque y Fairley, 2014).

En el proceso de desarrollo de requisitos se deben descubrir los requerimientos para desarrollar un sistema. Esto se logra comunicándose con las partes interesadas (clientes y usuarios del sistema) para obtener información sobre el dominio de la aplicación y los servicios que el sistema debe proporcionar, como el rendimiento del sistema, las limitaciones de *hardware*, etcétera (Sommerville y Sawyer, 1997).

Existe la posibilidad de que los requisitos sean transmitidos de manera incompleta o inconsistente debido a que los clientes utilizan el lenguaje natural para expresarse, mientras que los analistas prefieren un lenguaje más formal.

Con el fin de obtener resultados que aseguren la calidad del *software*, existen muchas propuestas de marcos de trabajo (*frameworks*) y lenguajes que coadyuvan a que los requisitos sean precisos, completos y consistentes (Yu, 1997).

El ideal es desarrollar el producto correcto en el primer intento, para lo cual es necesario detectar los errores en etapas tempranas del proyecto, cuando todavía se puede mitigar el riesgo de rehacer con mayor efectividad y a menor costo (figura 1). El método de ensayo y error, además de ser costoso y lento, crea insatisfacción en el cliente, por lo que se busca mejorar la calidad de los requerimientos para conseguir un impacto positivo dentro del proyecto en cuanto a costo, tiempo y nivel de satisfacción.

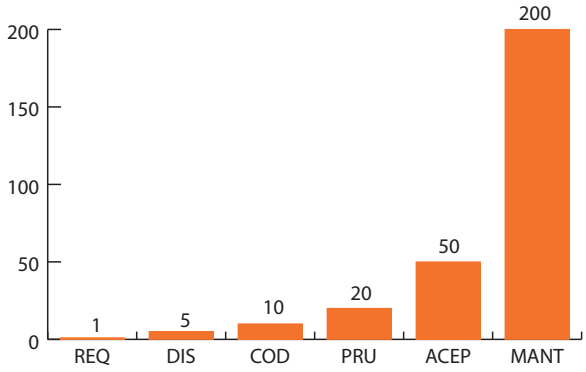

Figura 1. Costo de reparación de un error en función de la etapa en la que se detecta

Fuente: Sommerville y Sawyer, 1994

Se plantea en este artículo que a partir de un modelo gráfico orientado a objetivos basado en el lenguaje i*, es posible identificar los requisitos usando estereotipos para contextualizar los requerimientos de los usuarios. Un enfoque para lograrlo es la utilización de objetivos del negocio que expresen qué esperan los interesados del sistema. El ingeniero de requisitos debe traducirlos a tareas que representen la manera de alcanzarlos.

2.2 Metodología orientada a objetivos basada en i*

La metodología orientada a objetivos basada en i* plantea representar los requisitos a partir de un modelo gráfico, usando para ello estereotipos que contextualizan los objetivos del negocio en un lenguaje natural. Dichos objetivos de negocio expresan lo que los interesados esperan del sistema y deben ser traducidos por el ingeniero de requisitos en tareas que finalmente representan la manera de alcanzarlos.

González-Baixauli, Laguna y Leite (2004) refieren que esta metodología es utilizada para sentar las bases de la definición de los requerimientos de un *software*, donde cada estereotipo muestra un *goal* (meta) cuyo propósito es representar la funcionalidad del *software* a construir. Las tareas indican qué se debe hacer para alcanzar el *goal* y cada *softgoal* es un factor determinante para lograr un buen nivel de calidad en la definición del requerimiento en etapa temprana.

Los objetivos son operacionalizados en escenarios cuyas variantes están enfocadas en cada factor de calidad o *softgoal*. Las tareas implementan las acciones en los escenarios y están asociadas a diferentes contribuciones que provienen de cada *softgoal* (figura 2).

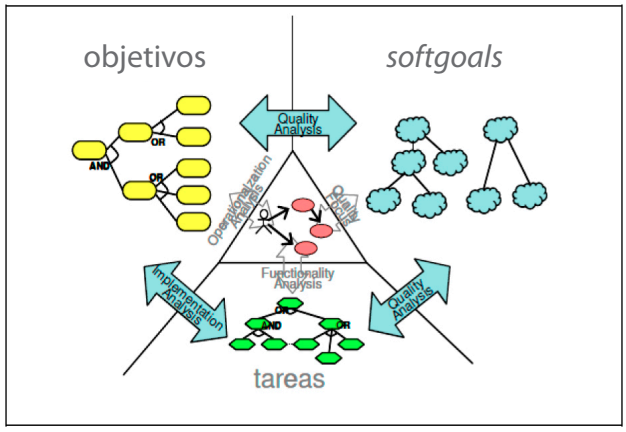

Figura 2. Relaciones entre modelos

Fuente: González-Baixauli, Laguna y Leite, 2004

Asimismo, en el contexto de la presente investigación se propone insertar información en los enlaces generados entre los elementos de esta metodología (*goal*, *softgoal*, tarea y recurso) a fin de gestionar la trazabilidad. La información adicional incluida debe permitir lo siguiente: a) identificar qué actores deben cumplir los objetivos y determinar el impacto de que alguno de éstos deje de implementarse; b) prever la contribución positiva o negativa de cada *softgoal* a fin de asegurar la calidad del *software*, y c) verificar la existencia de objetivos que no hayan sido identificados al inicio o que no hayan sido requeridos por el cliente.

2.3 Historias de usuario

Desde el punto de vista de las metodologías ágiles, las historias de usuario se construyen a partir de una conversación con el cliente. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea que se incluya en el diseño dicha funcionalidad (Cohn, 2009). También son utilizadas para la especificación de requisitos y es en este contexto en que se utilizará este término en la presente investigación.

En la figura 3, Moccia (2013) define el ciclo de especificación de requerimientos como una secuencia en la que el equipo encargado de requisitos plantea, en función de la estrategia y objetivos del negocio, los requerimientos que necesitan ser definidos y construidos durante la planificación de requisitos y la priorización por considerar.

De manera similar que un equipo de desarrollo, el equipo de requisitos debe planificar su *sprint*, planificar el trabajo y revisarlo. Si los requerimientos acopiados en el *requirements backlog* cumplen con las expectativas, pueden transferirlos al *product backlog* del equipo de desarrollo.

En muchos casos, las organizaciones tienen definidos documentos adicionales que deben crearse para pasar ciertos controles. Estos ítems deben considerarse en el *requeriments backlog*, pero no deben pasar al product *backlog*. No obstante, constituyen también material referencial para el equipo de desarrollo. En este punto es necesario señalar que la trazabilidad desde el *product backlog* hacia cualquier documento externo es una cuestión importante para favorecer la continuidad del proyecto.

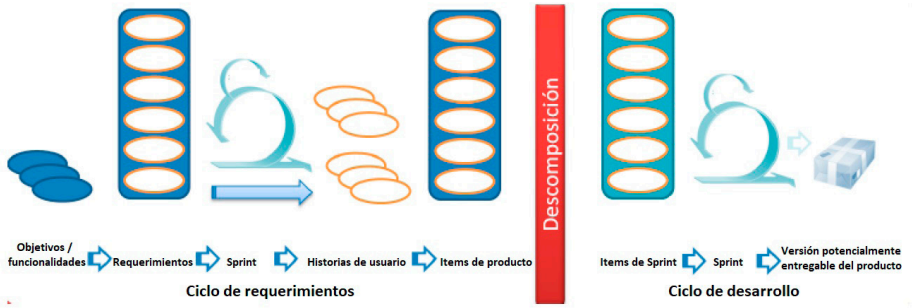

Figura 3. Definición y gestión de requerimientos ágiles

Fuente: Moccia (2014)

3. Herramienta de soporte

El proceso que debe cubrir la herramienta desarrollada ha seguido la metodología y la técnica mencionadas en el apartado anterior (figura 4). Este proceso toma como modelo el empleado por las metodologías ágiles para la especificación de requerimientos a través de historias de usuario y la técnica de modelado orientado a objetivos que utiliza el lenguaje de *framework* i*. Este modelado destaca las relaciones y la importancia de los objetivos de los interesados (*stakeholders*) junto con las contribuciones positivas o negativas de cada *softgoal* que permitan medir el impacto en la calidad del producto final.

Los objetivos representan los requisitos que se ejecutarán. Esta no es una decisión fácil porque cada objetivo podría recibir contribuciones de múltiples *softgoals* (requisitos no funcionales).

Las historias de usuario, generadas automáticamente, deben aprovechar toda esta información de manera que se evite la duplicación en el uso de técnicas de elicitación (entrevistas, cuestionarios, etcétera).

Este proceso descansa en una herramienta que será especificada teniendo en cuenta dos niveles: la arquitectura conceptual y el diseño técnico.

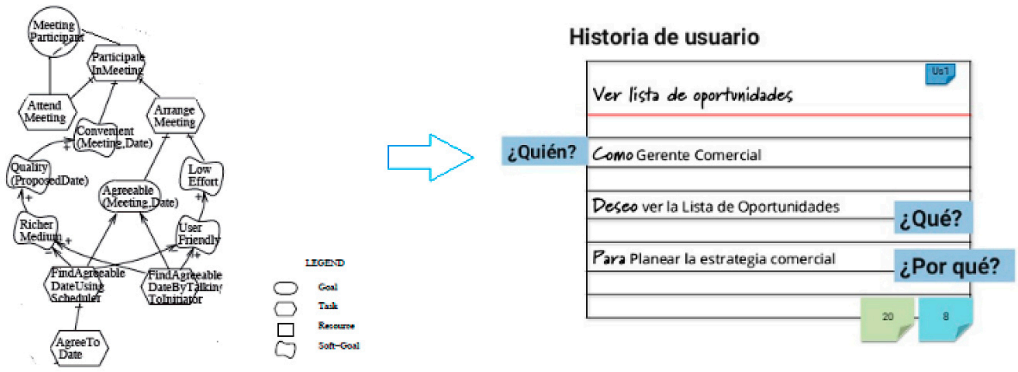

Figura 4. Proceso que debe cubrir la herramienta de soporte

Elaboración propia

3.1 Arquitectura conceptual

La arquitectura de MooTrace mostrada en la figura 5, utiliza el modelo arquitectónico Modelo-Vista-Controlador (MVC) el cual separa una aplicación en tres componentes principales: el modelo, la vista y el controlador. El marco de ASP.NET MVC proporciona una alternativa al modelo de formularios Web Forms de ASP.NET para crear aplicaciones *web*.

Según Microsoft (2018), ASP.NET MVC es un marco de baja complejidad y, al igual que las aplicaciones basadas en formularios Web Forms, se integra con las características de ASP.NET existentes, tales como páginas maestras y la autenticación basada en pertenencia. El marco MVC se define en el ensamblado System.Web.Mvc.

i. Modelo

Los objetos de modelo son las partes de la aplicación que implementan la lógica del dominio de datos de la aplicación. Frecuentemente los objetos de modelo recuperan y almacenan el estado del modelo en una base de datos.

ii. Vista

Las vistas son los componentes que muestra la interfaz de usuario de la aplicación. Normalmente esta interfaz de usuario se crea a partir de los datos de modelo.

iii. Controlador

Los controladores son los componentes que controlan la interacción del usuario, trabajan con el modelo y por último seleccionan una vista para representar la interfaz de usuario. En una aplicación MVC, la vista solo muestra información; el controlador administra y responde a los datos proporcionados por el usuario y su interacción.

Figura 5. Arquitectura conceptual de MooTrace

Elaboración propia

3.2 Diseño técnico de *MooTrace*

Después de la definición de la arquitectura conceptual de MooTrace, el siguiente paso consiste en seleccionar la tecnología que permitirá implementar dicha arquitectura, siguiendo la estructura mostrada en la figura 6.

Para el desarrollo de la aplicación se utilizó la base de datos Sql server 2014 y el entorno de desarrollo integrado Visual Studio en su versión 2015, que requiere del Framework 4.5.

El marco de desarrollo .NET reúne todos los componentes necesarios para el desarrollo de aplicaciones estables y escalables en entornos *web*. El componente que cumple estas condiciones es el ASP.NET, siendo el lenguaje de programación escogido el C#.

La vista será enriquecida con la librería *go.js*, la cual mediante el uso del elemento Canvas, provee un entorno para crear imágenes dinámicas en el que solo se requiere especificar las dimensiones. Adicionalmente, *go.js* permite crear gráficas complejas apoyándose en los distintos objetos que soporta. Tiene un nivel de flexibilidad bastante alto ya que, una vez creado el diagrama, este puede modificarse para añadir nuevos nodos al no estar construido con imágenes estáticas (Álvarez, 2014).

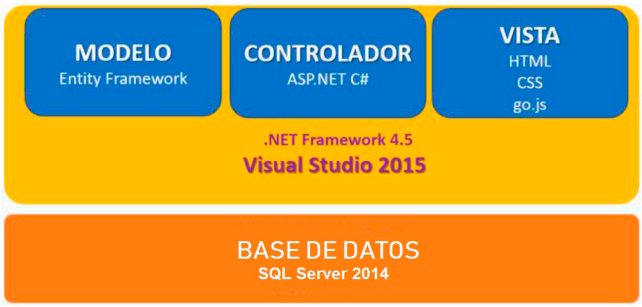

Figura 6. Diseño técnico de MooTrace

Elaboración propia

3.3 Implementación

En las secciones anteriores se han presentado los componentes que forman la estructura de la herramienta MooTrace y los medios tecnológicos a emplear para llevar a cabo su implementación. En esta sección se detalla la construcción de cada uno de ellos.

En la base de datos se creó la estructura basada en el enfoque del modelado orientado a objetivos.

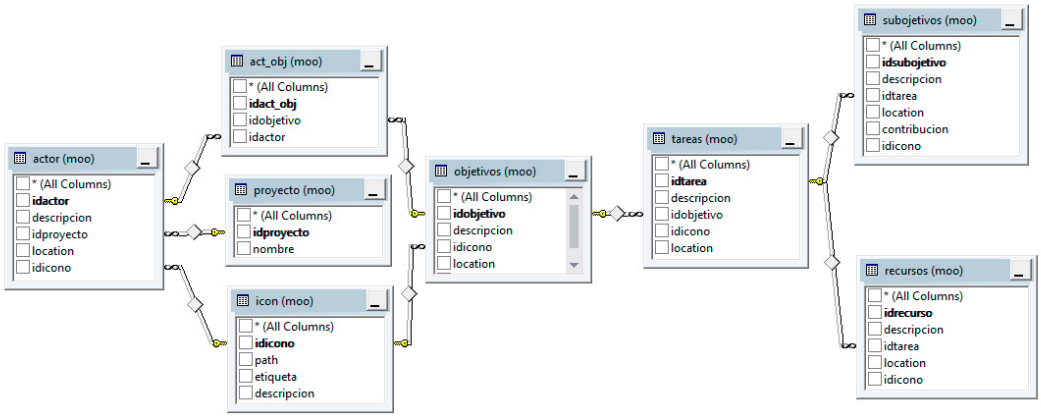

Figura 7. Modelo de la base de datos de MooTrace

Elaboración propia

Por otro lado, la codificación estuvo compuesta por cada uno de los elementos citados a continuación:

i. Modelo

Para trabajar con Entity Framework, se puede utilizar el esquema Database First, que consiste en generar mediante un asistente el modelo conceptual a partir de una base de datos existente, y obtener un archivo *.edmx con todos los objetos que se requiera mapear de la base de datos.

Se accede al menú emergente mediante clic derecho sobre la carpeta Models para agregar un nuevo elemento en la ventana; se selecciona Datos para poder visualizar el elemento ADO.NET Entity Data Model, al cual le asignamos el nombre de "modelmootrace". Luego se continúa haciendo las configuraciones correspondientes, obteniendo como resultado lo que se observa en la figura 8.

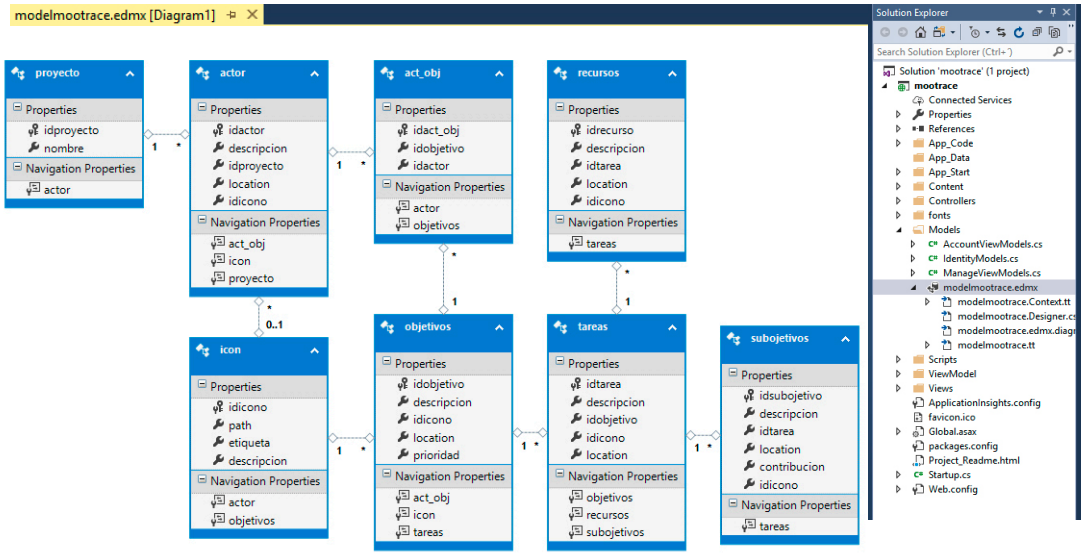

Figura 8. Modelo conceptual creado en la solución MooTrace

Elaboración propia

Cabe destacar que al expandir el modelo agregado en el explorador de soluciones (modelmootrace.edmx) se encontrarán dos plantillas (archivos de extensión *.tt). En la primera plantilla se almacena lo que se denomina contexto. A través de esta clase se establece la conexión con la base de datos. En la segunda plantilla se tienen las tablas que han sido mapeadas en el modelo conceptual (ver figura 9).

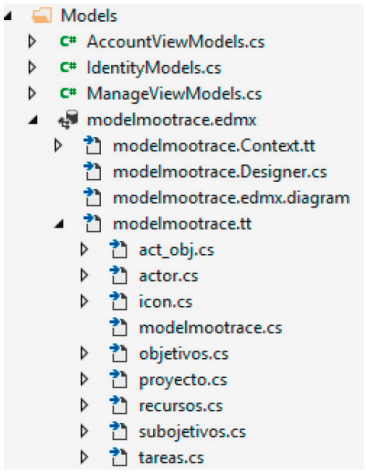

Figura 9. Archivos de mapeo generados a partir del modelmootrace.edmx

Elaboración propia

ii. Vista

En esta carpeta se encuentran las vistas generadas a partir de los controladores, las cuales devolverán y enviarán datos desde el cliente *web*.

Cabe destacar que desde la vista de ModeloMoo se referencia a las librerías para una presentación más dinámica de los estereotipos del modelo gráfico, a través de las siguientes etiquetas (*tags*).

```
<script src="~/Content/js/go.js"></script>
<script src="~/Content/js/go-debug.js"></script>
```

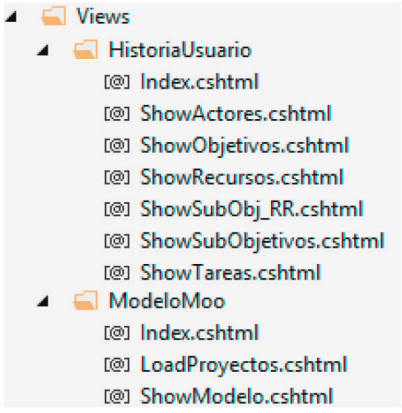

Figura. 10. Vistas creadas en la solución MooTrace

Elaboración propia

iii. Controlador

Hemos creado dos archivos de tipo *controller* que llaman a los datos a través del modelo para ser mostrados en la vista, como se muestra en la figura 11.

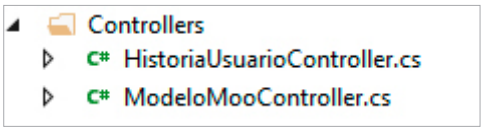

Figura 11. Controladores creados en la solución MooTrace

Elaboración propia

El componente ModeloMooController.cs es el responsable de procesar las solicitudes de entrada, manipular la entrada, guardar datos y enviar una respuesta de vuelta al usuario respecto al diagrama del modelo orientado a objetivos.

El componente HistoriaUsuarioController.cs es el responsable de procesar las solicitudes de entrada, manipular la entrada, guardar datos y enviar una respuesta de vuelta al usuario, respecto a la historia de usuario generada a partir del controlador ModeloMooController.cs.

3.4 Implementación de un caso de estudio

El caso de estudio se basó en métricas que permitieron una evaluación cuantitativa de la herramienta *MooTrace* que incluyó los factores propuestos por el estándar ISO 9126.

El perfil de los 70 participantes de las pruebas se caracterizó por tener una mediana de ocho años de experiencia en análisis y diseño de sistemas. El 40 % de ellos tenía menos de cinco años de experiencia y el 100 % de ellos estaba asignado a un proyecto. Fue posible subagruparlos por proyecto asignado.

El objetivo de las pruebas fue medir el nivel de usabilidad de la herramienta MooTrace mediante la realización de las siguientes tareas: a) elaborar un modelo gráfico por parte del grupo de analistas, y b) generar de manera automatizada el formato de historia de usuario por parte del grupo de desarrolladores.

La herramienta fue probada por dos grupos, el primero de 30 analistas y el segundo de 40 desarrolladores, a los que se impartió una inducción acerca de los estereotipos para la creación del modelo gráfico que plantea el enfoque propuesto (figura 12). El primer día de pruebas se trabajó con los analistas y el segundo día de pruebas con los desarrolladores.

Figura. 12. Estereotipos de la solución MooTrace

Elaboración propia

Al completar el proceso de prueba, los participantes resolvieron un cuestionario de ocho preguntas referido al uso de la herramienta y se les solicitaron sugerencias de mejora. En la figura 13 se puede observar un ejemplo del modelo gráfico creado con la herramienta MooTrace, y en la figura 14 se aprecia un ejemplo de la historia de usuario generada a partir del paso anterior en siete segundos. Este es un valor referencial.

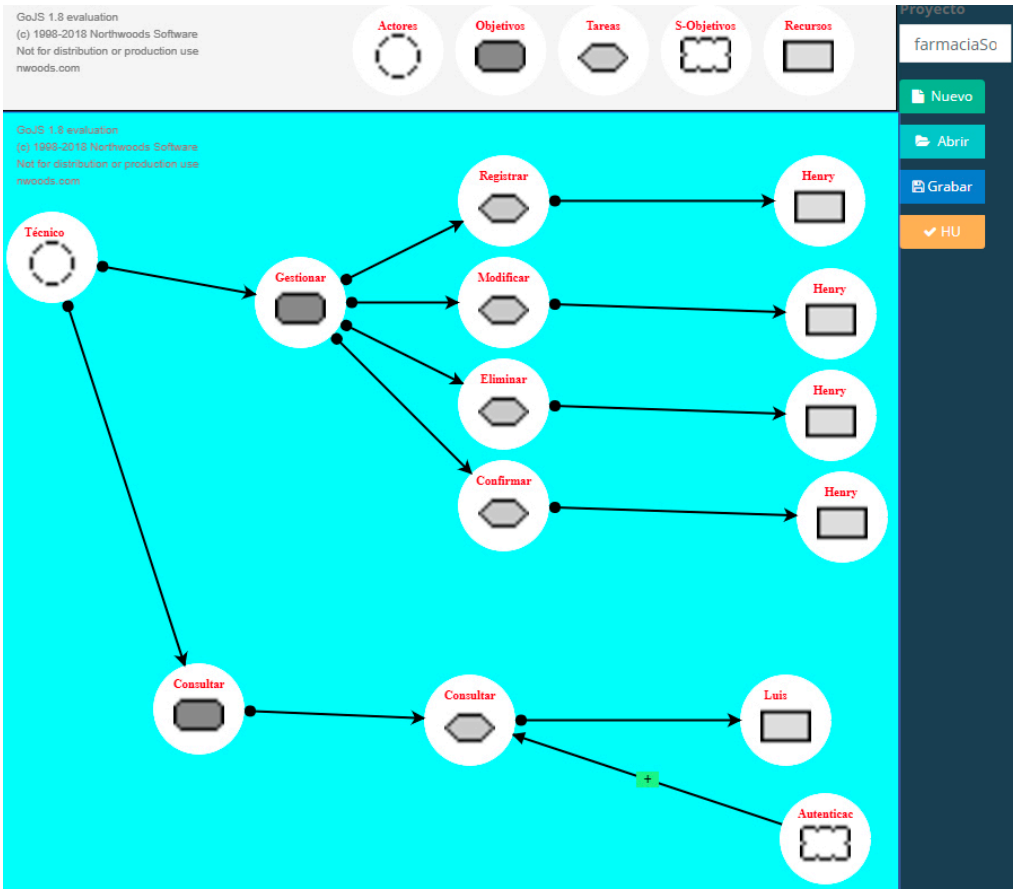

Figura. 13. Creación del modelo gráfico

Elaboración propia

Figura 14. Automatización de las historias de usuario

Elaboración propia

4. Resultados

Los indicadores de satisfacción más representativos obtenidos a partir del proceso de pruebas se presentan en la tabla 1.

Tabla 1. *Principales indicadores de usabilidad de MooTrace*

Pregunta del cuestionario	Indicador
¿Los estereotipos usados en el modelo gráfico son entendibles?	El 90 % de los encuestados afirma que entendió rápidamente el propósito de cada estereotipo del modelo gráfico.
Al crear el modelo gráfico, ¿representa de manera sencilla los requerimientos esperados por el cliente?	El 85 % de los encuestados afirma que el modelo representa lo requerido, aunque esperan un mayor detalle.
Al automatizar la creación de las historias de usuario, ¿considera que el tiempo ahorrado es relevante?	El 100 % de los encuestados afirma que es importante el ahorro de tiempo en la automatización de este formato.

Elaboración propia

Asimismo, los participantes en las pruebas expresaron tener expectativas en que la información generada por MooTrace contribuya en la gestión de la trazabilidad de los requerimientos y sirva también para analizar el impacto del cumplimiento y de los cambios en las tareas o recursos considerados en el modelo gráfico.

5. Conclusiones y recomendaciones

La ingeniería de requisitos tiene un papel cada vez de mayor relevancia como garantía del cumplimiento eficaz y eficiente del proceso de desarrollo de *software*. Por ello, cualquier indicador, incidencia o factor de éxito en el proceso de desarrollo de requisitos debe ser oportunamente identificado para ser aprovechado adecuadamente. En este contexto y a partir de las pruebas desarrolladas con la herramienta *MooTrace* la presente investigación establece las siguientes conclusiones:

La herramienta de modelado gráfico propuesta permitió representar con suficiencia los requisitos, de acuerdo con lo manifestado por el 85 % de analistas participantes en las pruebas, siendo deseable alcanzar un mayor nivel de detalle.

El rol de los estereotipos de la herramienta de modelado gráfico, así como su propósito y contribución a la comprensión del requerimiento, fueron reconocidos como adecuados por el 90 % de analistas que participaron en las pruebas.

Se comprueba la factibilidad de generar un formato de historia de usuario a partir del modelado gráfico en un tiempo referencial de siete segundos, lo cual es apreciado como un ahorro relevante de tiempo en las tareas de refinamiento de requisitos por el 100 % de los desarrolladores participantes.

La mejora de esta herramienta de modelado gráfico orientada a objetivos debe cubrir en el futuro la totalidad del ciclo de gestión de los requerimientos, lo cual permitirá tener perspectivas distintas para cada miembro del equipo sin vulnerar el origen del requisito en sí. Queda como propuesta gestionar la trazabilidad de los requerimientos, así como analizar el impacto del cumplimiento y de los cambios en las tareas o recursos del modelo gráfico.

Referencias

Álvarez, C. (2014), *Go JS interactive diagrams for the web*. Genbeta. Recuperado de https://www.genbeta.com/desarrollo/go-js-una-libreria-para-html5-canvas.

Bourque, P. y Fairley, R. E. (eds.) (2014). *Guide to the software engineering body of knowledge (SWEBOK(R))*: Version 3.0. Piscataway, N. J.: IEEE Computer Society Press.

Cohn, M. (2009). *User stories applied: For agile software development.* Boston: Pearson Education.

González-Baixauli, B., Laguna, M. A. y Leite, J. C. S. (2004). Análisis de variabilidad con modelos de objetivos. En M. Ridao y L. Marcio Cysneiros (eds.). *Workshop em Engenharia de Requisitos* (WER). Tandil, Argentina, (pp. 77-87).

Microsoft (2018). Información general sobre ASP.NET MVC. *Developer Network*. Recuperado de https://msdn.microsoft.com/es-es/library/dd381412(v=vs.108).aspx.

Moccia, J. (2013). *Agile requirements, definition and management (RDM)*. OneSpring. Recuperado de http://community.protoshare.com/wp-content/uploads/2014/03/Agile_RDM.pdf.

Sommerville, I. y Sawyer, P. (1997). *Requirements engineering: a good practice guide*. John Wiley & Sons, Inc.

Yu, E. S. (1997). Towards modelling and reasoning support for early-phase requirements engineering. En *Proceedings of the Third 1997 IEEE International Symposium, Requirements Engineering* (pp. 226-235). Annapolis, MD, USA: Institute of Electrical and Electronicals Enginers.