Un Modelo Conceptual para la Especificación y Trazabilidad de Requerimientos Funcionales basados en Casos de Uso y Casos de Prueba

M. Luciana Roldán[1], Marcela Vegetti[1], Marcelo Marciszack[2], Silvio Gonnet[1], Horacio Leone[1]

1.- Instituto de Desarrollo y Diseño (Conicet/UTN)
Universidad Tecnológica Nacional, Fac. Regional Santa Fe
{lroldan, mvegetti, sgonnet, hleone}@santafe-conicet.gov.ar

2.- Centro de Investigación, Desarrollo y Transferencia de Sistemas de Información (CIDS)
Universidad Tecnológica Nacional, Fac. Regional Córdoba
marciszack@gmail.com

Resumen

Existe una estrecha relación entre las actividades de la ingeniería de requerimientos y las pruebas de sistemas de software intensivos. Por un lado, una especificación completa, consistente y legible de requerimientos permite una buena definición de casos de prueba. Por otro lado, la realización de actividades de prueba, particularmente la definición de los casos de prueba, provee información valiosa para la mejora de la especificación de requerimientos. En este trabajo se propone un enfoque basado en modelos de prueba que permite la derivación de casos de prueba (CP) a partir de la especificación de casos de uso (CU) textuales, que permite definir la cobertura de las pruebas a realizar sobre el sistema. Tal derivación se logra mediante un artefacto intermedio denominado modelo de prueba, el cual tiene una doble función. Por un lado permite validar posibles cursos de acción en un CU, y por el otro permite generar CP teniendo en cuenta diferentes criterios de cobertura sobre el comportamiento especificado en los CU. Para alcanzar el objetivo se construye una ontología basada en un modelo conceptual que permite definir e integrar conceptos relativos a los metamodelos de las diferentes herramientas de soporte a las actividades del proceso de desarrollo de software intervinientes, posibilitando la interoperabilidad entre ellas para lograr consistencia y trazabilidad de artefactos.

1. Introducción

Los requerimientos de un sistema describen lo que el sistema debe realizar, los servicios que debe proveer y las restricciones de su operación. Son una pieza fundamental en un proyecto de desarrollo de software, ya que marcan su punto de partida y permiten verificar si los objetivos establecidos en el proyecto fueron alcanzados. Una adecuada especificación de requerimientos contribuye al desarrollo de software de mejor calidad, ahorro de tiempo y dinero, y minimiza el riesgo de exceder presupuestos y el fracaso de los proyectos de desarrollo [14]. Es necesario, por lo tanto, generar especificaciones correctas que describan con claridad, sin ambigüedades, en forma consistente y compacta, las necesidades de usuarios y clientes.

Existe una estrecha relación, recíproca y positiva, entre las actividades de la ingeniería de requerimientos y las pruebas de sistemas de software intensivos. Por un lado, una especificación completa, consistente, y legible de requerimientos soporta la definición de casos de prueba. Por el otro lado, la realización de actividades de prueba, particularmente la definición de los casos de prueba, provee información valiosa para la mejora de la especificación de requerimientos. Es sabido que en muchas organizaciones y empresas de desarrollo de software, el costo de la etapa de pruebas dentro del proceso de desarrollo de software es muy alto [18], considerando tanto la generación como la ejecución de los casos de prueba. Adicionalmente, existen altos costos en los proyectos por retrabajo, debido a la aparición de incidencias en etapa de mantenimiento por deficiencias que podrían haberse evitado en etapas tempranas y realizando las pruebas pertinentes. Dado que en general, la mayoría de los procesos de desarrollo de software tienen un enfoque iterativo, es recomendable comenzar a realizar las pruebas lo antes posible, al mismo tiempo que se va pasando por las distintas fases del ciclo de vida del software. Siguiendo esta línea, el presente trabajo se alinea a las propuestas de iniciar el proceso de pruebas en forma temprana, utilizando los casos de uso propuestos durante la etapa de requerimientos para generar los casos de prueba.

Los casos de uso son claves para el proceso de prueba, no solo porque de ellos se derivan los demás artefactos, sino también porque identifican y establecen las condiciones que deberán tenerse en cuenta al

implementar los casos de prueba, así como también son necesarios para verificar que se han implementados satisfactoriamente y con calidad todos los requisitos que debe cumplir el sistema resultante. Por lo tanto, son necesarias estrategias y técnicas que brinden soporte a la especificación de requerimientos funcionales y la generación de casos de prueba a partir de éstos, de tal manera que faciliten su validación y trazabilidad.

El objetivo de este trabajo es partir de la definición de requerimientos funcionales mediante plantillas de casos de uso textuales, generar casos de prueba en forma sistemática a partir de tales casos de uso, y lograr trazabilidad entre estos artefactos, a fin de alcanzar un producto de software de calidad. La derivación de casos de prueba a partir de casos de uso se logra mediante un artefacto intermedio, denominado modelo de prueba, el cual tiene una doble función. Por un lado, permite validar posibles cursos de acción en un caso de uso, y por el otro, permite la derivación de casos de prueba teniendo en cuenta diferentes criterios de cobertura sobre el comportamiento especificado en los requerimientos funcionales mediante casos de uso.

Para alcanzar el objetivo se construye una ontología de requerimientos funcionales basados en casos de uso, que posibilita la interoperabilidad semántica entre las diversas herramientas que pueden estar involucradas en el proceso de especificación de casos de uso, su validación, y la derivación de casos de pruebas trazables hacia los requerimientos.

El resto del trabajo se estructura de la siguiente manera. En la sección 2 se analiza en primer lugar los conceptos fundamentales de la propuesta. Luego, se define incrementalmente el modelo conceptual y las estrategias propuestas para la definición de casos de uso, y para la derivación de casos de prueba a partir de modelos basados en autómatas finitos. En la sección 3, se describe el proceso de construcción de la ontología. En la sección 4, se describen trabajos relacionados. Finalmente, en la sección 5 se presentan las conclusiones y trabajos a futuro.

2. Modelo conceptual de requerimientos funcionales, casos de uso y casos de pruebas

El enfoque propuesto propone la construcción de una ontología de requerimientos funcionales que posibilite la interoperabilidad entre herramientas utilizadas en la especificación de casos de uso (HECU) y herramientas de gestión de casos de pruebas (HGCP). La construcción de dicha ontología implica la identificación de los conceptos fundamentales y sus relaciones. A continuación se irán abordando cada uno de éstos, de manera de incrementalmente definir un modelo conceptual que servirá como base para la posterior identificación de términos a incluir en la ontología. El modelo conceptual debe posibilitar la integración de metamodelos de diferentes herramientas HECU y HGCP.

2.1. Conceptos fundamentales

Los casos de uso describen una secuencia de interacciones que un actor puede llevar a cabo con un sistema para lograr algún resultado de valor [20]. Un caso de uso podría abarcar cierta cantidad de actividades relacionadas que tienen un objetivo o meta común. Una definición similar es la de Pohl [14], en donde un caso de uso se define como la especificación de acciones, incluyendo secuencias, variantes, y secuencias de error, que el sistema, subsistema, o clase puede realizar interactuando con objetos externos para proveer un servicio de valor. En otras palabras, un caso de uso es una colección de escenarios de uso relacionados, lo que implica además que un escenario es una instancia específica de un caso de uso en la cual se atraviesa, un curso de acción posible.

La forma más comúnmente empleada para la especificación de los casos de uso es mediante descripciones textuales utilizando plantillas estructuradas para una definición detallada. En la Figura 1 se presenta un modelo que describe los principales conceptos intervinientes en la especificación de un caso de uso, el cual está basado en la plantilla de caso de uso propuesta por Pohl [14]. Los objetivos (*Goal*) representan las intenciones de los "stakeholders". Los escenarios (*UseCaseScenario*) documentan las secuencias de interacciones por medio de las cuales se satisfacen o no tales objetivos. Los casos de uso (*UseCase*) agregan múltiples escenarios que están asociados al mismo objetivo u objetivos. Según se indica en la Figura 1, se pueden tener diferentes escenarios en un caso de uso. Un escenario principal (*MainScenario*) documenta la secuencia de interacciones o pasos (*ScenarioStep*) que es normalmente ejecutada a fin de satisfacer un objetivo dado. Por otro lado, un caso de uso puede tener uno más escenarios alternativos (*AlternativeScenario*). Un escenario alternativo documenta una secuencia de pasos que puede ser ejecutada en lugar del escenario principal y que resulta en la satisfacción de los objetivos asociados al escenario principal. En cuanto a los escenarios de excepción (*ExceptionScenario*), éstos documentan una secuencia de interacciones que se ejecuta en lugar de cualquier otro escenario (principal, alternativo, o excepción), a causa de la ocurrencia de un evento excepcional. Esto implica, que los objetivos asociados con el escenario principal no pueden ser satisfechos.

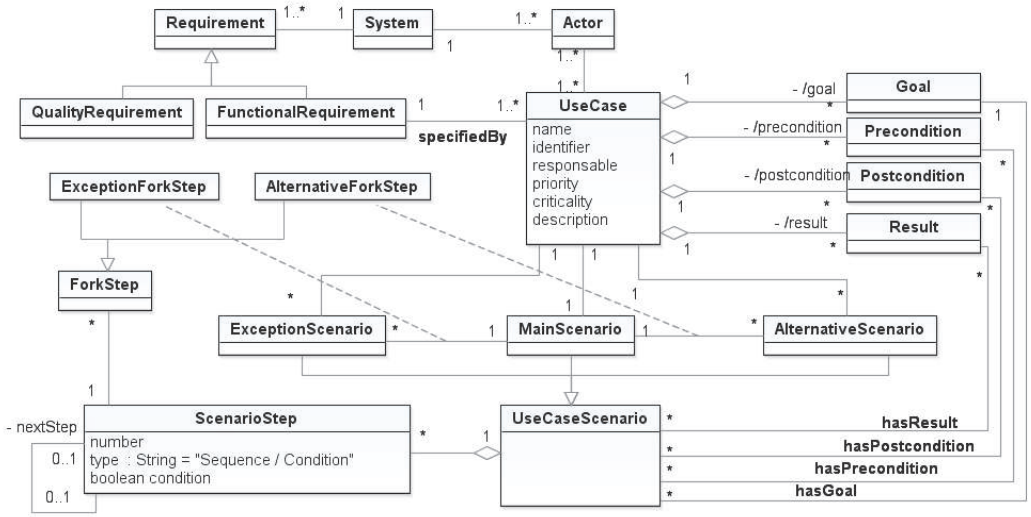

Figura 1. Modelo conceptual de casos de usos y escenarios.

Dentro del proceso de desarrollo de un software, los casos de uso representan una primera definición de los requerimientos funcionales que los desarrolladores deben implementar. Éstos conducen a la definición de los casos de prueba que deben realizarse para verificar que el software cumple con los requerimientos planteados.

Un caso de prueba (*TestCase*) especifica la información requerida para la ejecución de una prueba. Un caso de prueba comprende las precondiciones requeridas para la ejecución de la prueba (*TestPreconditions*), el conjunto de entradas y salidas esperadas (*TestInput* y *TestExpectedOutput*), las instrucciones para la prueba (cómo las entradas son pasadas al objeto de prueba y cómo las salidas son leídas desde el objeto de prueba) y las postcondiciones esperadas (*TestExpectedPostconditions*) [14]. Esta definición puede ser plasmada en el modelo conceptual presentado en la Figura 2. Un caso de prueba se ejecuta pasando la entrada especificada en el caso de prueba al objeto de prueba (*TestObject*). Por lo tanto, la ejecución de un caso de prueba comprende una secuencia de interacciones entre los elementos en el contexto de la prueba (por ejemplo los testers o usuarios) y el objeto de la prueba. Al momento de definir un caso de prueba, las interacciones correspondientes pueden entonces ser definidas por medio de escenarios a diferentes niveles de abstracción. Pohl [14] define el concepto de escenario de caso de prueba (*TestCaseScenario*), para hacer referencia a "tipos de casos de prueba", los cuales describen a conjuntos de casos de prueba que tienen cierto tipo de entradas y ciertos tipos de salidas esperadas (*TestInputType* y *TestExpectedOutputType*, respectivamente).

2.2. Transformación de casos de usos en máquina de estados

En trabajos previos, se ha contribuido en la caracterización de modelos para la especificación de requerimientos funcionales. En esta línea, se propuso un método que permite la trazabilidad y validación de requerimientos funcionales de un sistema de información mediante la transformación de modelos conceptuales [10, 11, 12]. Tal método fue implementado en una herramienta de software denominada SIAR (Sistema Integral de Administración de Requerimientos) que brinda soporte a la administración de requerimientos funcionales y utiliza casos de uso UML (Lenguaje Unificado de Modelado) para su representación. SIAR es una herramienta que permite el registro de requerimientos funcionales de manera sencilla, posibilita la trazabilidad de cambios y lleva a cabo validaciones funcionales sobre los mismo. Una de las funcionalidades provistas por la herramienta es un procedimiento

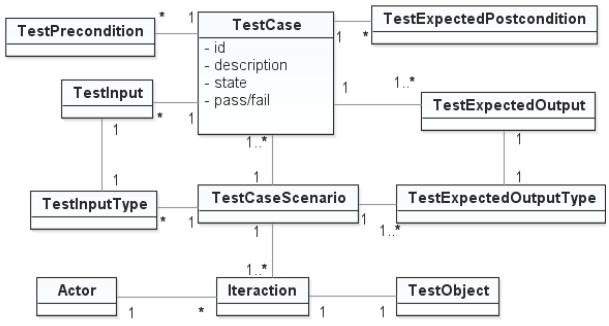

Figura 2. Modelo conceptual de casos de pruebas.

automatizado de análisis de consistencia de Casos de Uso, a través de la verificación de secuencias de acciones descriptas en el caso de uso. Dicho procedimiento, está enfocado en la especificación y análisis de requerimientos desde el punto de vista de comportamiento, para lo cual se emplean autómatas finitos deterministas. En este sentido, el sistema genera el grafo que representa las transiciones de estados de cada Caso de Uso especificado. Luego, un simulador de autómatas finitos deterministas lleva a cabo un análisis para verificar la cohesión de los escenarios en él definidos.

Los autómatas finitos deterministas (AFD, o DFA de por sigla en inglés) [8] son frecuentemente empleados para documentar el comportamiento reactivo de un sistema. Formalmente, un autómata finito determinista se define como una quíntupla (Q, Σ, δ, q_0, F), donde sus componentes son:

1. Un conjunto finito de estados (Q).
2. Un conjunto finito de símbolos de entrada (Σ).
3. Una función de transición que toma como argumentos un estado (de Q) y un símbolo de entrada (de Σ) y devuelve un estado (del conjunto Q). La función de transición se designa habitualmente como δ. Si q es un estado y a es un símbolo de entrada, entonces $\delta(q, a)$ es el estado p tal que existe un arco etiquetado a que va desde q hasta p.
4. Un estado inicial q_0, uno de los estados de Q.
5. Un conjunto de estados finales o de aceptación F, donde F es un subconjunto de Q.

Además de la representación matemática como quíntupla, un AFD puede ser representado como un grafo llamado diagrama de transiciones, en donde: a) para cada estado q de Q, existe un nodo; b) para cada estado q de Q y cada símbolo de entrada a de Σ, sea $\delta(q, a) = p$, el diagrama de transiciones tiene un arco desde el nodo q hasta el nodo p, etiquetado como a; c) existe una flecha dirigida al estado inicial q_0 que no tiene origen en ningún nodo, que representa el estado de inicio; d) los nodos correspondientes a los estados de aceptación (los que pertenecen a F) están marcados con un doble círculo. Los estados que no pertenecen a F tienen un círculo simple. La definición de AFD se representa en la Figura 3, con un modelo conceptual donde se explicitan sus componentes y las relaciones entre ellos.

Para transformar un caso de uso en una máquina de estados de este tipo, se definió un conjunto de reglas de conversión que permiten obtener un AFD a partir de un caso de uso [10]. En el presente trabajo, esas reglas fueron redefinidas para superar algunas limitaciones que poseía el enfoque original, el cual no era compatible con la transformación de casos de usos con bifurcaciones por escenarios alternativos múltiples.

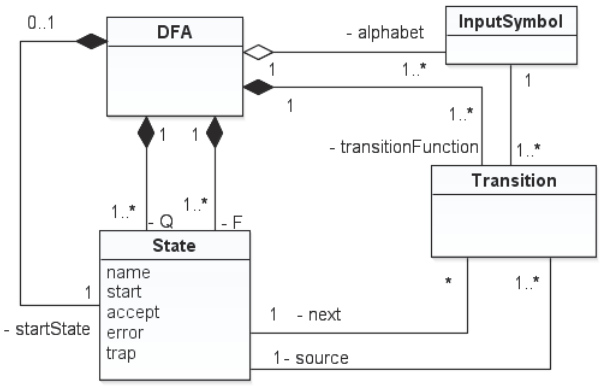

Figura 3. Modelo conceptual de AFD.

Las reglas de transformación se enuncian a continuación:

R1. Cada paso de un curso de acción de un caso de uso (en cualquiera de sus escenarios) responde a un estado y es un nodo de la máquina de estados. Las condiciones (paso inicial de un escenario alternativo o excepción) se consideran también pasos que tienen su estado correspondiente en el AFD.

R2. Todo caso de uso tiene un paso inicial, el cual es el primer paso de todos los cursos de acción posibles (o flujos) y está dado por el primer paso del escenario principal. Este paso constituye el estado/nodo inicial del AFD generado.

R3. Todo caso de uso tiene un paso final en su curso de acción normal dado (escenario principal). Este último paso constituye uno de los posibles estados finales del AFD (pertenece al conjunto F). Este estado final significa la satisfacción del objetivo del caso de uso.

R4. Un curso de acción en caso de uso puede no tener, tener uno o tener varios estados finales por error, siendo cada uno de éstos el último paso de un escenario de excepción. Estos pasos son también estados/nodos finales del AFD.

R5. Toda secuencia posible entre un paso de escenario a otro paso de escenario, define una transición del AFD que representa al caso de uso.

R6. Si en cierto paso del escenario principal del caso de uso, existe una bifurcación posible hacia un escenario alternativo, se generará una transición saliente desde el estado correspondiente a dicho paso hacia otro estado que represente el primer paso del escenario alternativo correspondiente. Esto significa que partiendo de un estado/nodo origen se puede continuar a un único estado/nodo destino, o a dos nodos/estados destino.

R7. El grafo de estados asociado al caso de uso tiene un alfabeto de dos símbolos (N, A) para indicar qué evento provoca el paso de un estado/nodo a otro:

- N: La transición se produce continuando con el curso de acción determinado por el escenario actual.

- *A*: Se presenta una condición que bifurca el curso de acción actual a un escenario alternativo o de excepción. Si la condición dada por el primer paso del escenario alternativo o de excepción se cumple, se continúa con el curso de acción indicado por el escenario (evento dado por un símbolo N). Si la condición no se verifica y existe otro escenario alternativo para la bifurcación, se considera que ocurre nuevamente un evento dado por A, y se provoca una transición hacia el primer paso de tal escenario. Así se continúa sucesivamente hasta que no existan más alternativas para un mismo paso.

De esta manera, queda determinado el alfabeto del AFD como el conjunto $\Sigma = \{N, A\}$.

Por lo tanto, todo curso de acción válido en un escenario de un caso de uso, está dado por una secuencia de símbolos de entrada, para la cual, partiendo del estado de inicio (primer paso en el escenario principal), es posible alcanzar un estado final perteneciente a F, dado por el estado final del escenario principal o un estado final de un escenario de excepción.

Para ilustrar cómo se genera un AFD a partir de un caso de uso a partir de las reglas de transformación definidas, se presenta un ejemplo de caso de uso en la Figura 4. La plantilla empleada para su especificación está basada en las propuestas por Cockburn [4] y Pohl [14], la cual se encuentra implementada en el sistema SIAR para normalizar la carga de casos de uso. Este caso de uso describe las interacciones que ocurren cuando un usuario accede a un sistema web (login).

En la Figura 5 (lado a), se muestra el AFD que modela el comportamiento del caso de uso integrando todos los escenarios definidos en el mismo. Para obtenerlo se aplicaron las reglas definidas previamente. La definición formal del caso de uso es *CU-01 = (Q, Σ, δ ,1, F),* donde $\sum = \{ N, A \}$, *Q = {1, 2, 3, 4, 3a, 3a1, 3a2, 3a3, 3b, 3b1, 3b2, 3b3, 3b4, 3b5, 4a, 4a1, 4a2, 4b, 4b1, TrapState}*, *F = {4, 4b1}*.

Cabe aclarar que para simplificar la visualización el grafo, no se muestra el AFD en forma completa en el sentido estricto, obviando la visualización del estado trampa (*TrapState*). Esto implica que en el grafo no están presentes todas las transiciones para cada uno de los símbolos del alfabeto N y A por cada estado.

Una vez que se carga el AFD obtenido en el simulador, es posible ingresar diferentes cadenas de entradas, que representan los posibles escenarios del caso de uso a fin de comprobar si éstas son aceptadas o rechazadas. Dado el alfabeto definido para el autómata equivalente al caso de uso, la entrada a verificar será una cadena formada por los símbolos "N" y "A".

ID Caso de Uso	CU-01		
Nombre	Acceso al sistema		
Versión	V.1.0		
Descripción	Un usuario desea acceder al sistema, para lo cual debe hacer "login" con su usuario y contraseña. Si no cuenta con un usuario, deberá registrarse para crear su cuenta en el sistema.		
Actores	Usuario		
Objetivo	Acceder al sistema y ser autenticado		
Precondiciones	El usuario debe estar registrado en el sistema		
Postcondiciones	El usuario ha ingresado al sistema. Se registró el acceso del usuario en la tabla de actividad.		
Resultado	El usuario ingresa a sistema y puede permanecer allí hasta desloguearse o pase cierto tiempo de inactividad.		
Escenario principal	1	El usuario accede al sistema a través de un navegador ingresando la URL	
	2	El usuario ingresa a la opción de menú de login	
	3	El usuario proporciona su nombre de usuario y contraseña	
	4	El sistema valida las credenciales del usuario y le da la bienvenida	
Escenarios alternativos	3a	El usuario no recuerda su contraseña	
		3a1	El usuario solicita el envío de contraseña por email
		3a2	El sistema envía una clave temporal al email que proporcionó el usuario al momento de registrarse
		3a3	Continúa al paso 3
	3b	El usuario no está registrado en el sistema	
		3b1	El usuario solicita crear una cuenta
		3b2	El sistema solicita los datos para crear la cuenta
		3b3	El usuario ingresa los datos requeridos y opcionales solicitados y confirma
		3b4	El sistema crea la cuenta del usuario
		3b5	Continúa al paso 2
	4a	Contraseña incorrecta. Intento < 3	
		4a1	El sistema da un mensaje al usuario indicando que la contraseña es incorrecta.
		4a2	Continúa al paso 2
Escenarios de excepción	4b	Contraseña incorrecta. Intento = 3	
		4b1	El sistema da un mensaje y bloquea la cuenta por seguridad.

Figura 4. Caso de uso CU-01.

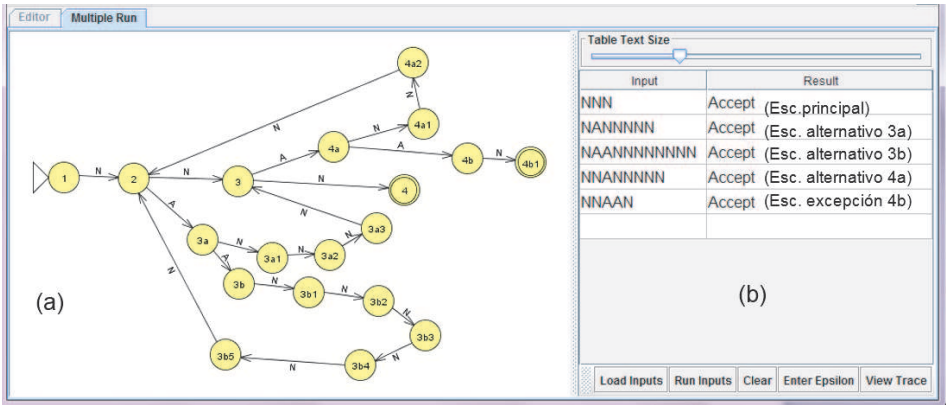

Figura 5. (a) AFD correspondiente al CU-01. (b) Simulación de escenarios para el CU-01.

Si la cadena es aceptada, significa que el escenario está correctamente definido en dicho caso de uso. Por el contrario, si la cadena es rechazada, significa que en algunos de los caminos o escenarios hay un error en la secuencia de pasos. Esto puede deberse a que:

- hay estados inconexos, caminos o pasos que no fueron considerados al definir el caso de uso,
- se definieron escenarios que no finalizan adecuadamente quedando inconclusos, o
- existen bucles infinitos en la definición de escenarios.

Luego, el error puede ser corregido en una nueva versión del caso de uso, generando su correspondiente AFD automáticamente. De esta manera se logran validar los casos de uso, y se establece trazabilidad entre las versiones de escenarios que conforman un caso de uso que describen a un requerimiento funcional. En lado (b) de la Figura 5, se muestra la validación de los cinco escenarios posibles definidos en el caso de uso *CU-01*. Para la comprobación de escenarios se trabajó con la herramienta Jflap (www.jflap.org), la cual permite la especificación de archivos en formato XML.

2.3. Derivación de casos de prueba a partir de casos de uso

Diversos autores han propuesto estrategias para la derivación de casos de pruebas basadas en requerimientos. Mientras que unos plantean la posibilidad de derivarlos directamente a partir de artefactos de requerimientos (como podrán ser una SRS, diagramas de casos de uso, o casos de usos textuales estructurados) [1], otros proponen su derivación a partir de los llamados "modelos de pruebas" (Model-based test-case derivation) [14, 15]. En este último caso, un modelo de prueba sirve como la base para derivar sistemáticamente casos de pruebas. Técnicas tales como las máquinas de estados (autómatas finitos, diagramas de estados UML, diagramas de actividades UML), y diagramas de flujos son adecuadas para crear los modelos de prueba. Las relaciones entre casos de uso y casos de pruebas mediante modelos de prueba se presentan en la Figura 6, adaptando la propuesta de Pohl [14]. Como se describió en la Figura 1, un caso de uso contiene uno o varios escenarios de caso de uso asociados. El modelo de prueba es desarrollado en base a los escenarios de casos de uso. Los escenarios de casos de prueba individuales se identifican en base al modelo de prueba generado. Finalmente, los casos de prueba se derivan a partir de los escenarios de casos de prueba, especificando valores concretos para las entradas y resultados esperados.

El método propuesto para obtener un AFD a partir de un caso de uso con el objetivo de obtener un modelo de comportamiento que sirva para validar la consistencia de escenarios, también puede ser utilizado como "modelo de prueba" para la derivación de casos de prueba de tipo de aceptación, de manera que sea posible generar todos los flujos sobre el caso de uso que se desean probar. Para reflejar esta idea, se incorpora en el modelo conceptual de la Figura 6 una relación de generalización/especialización entre los conceptos

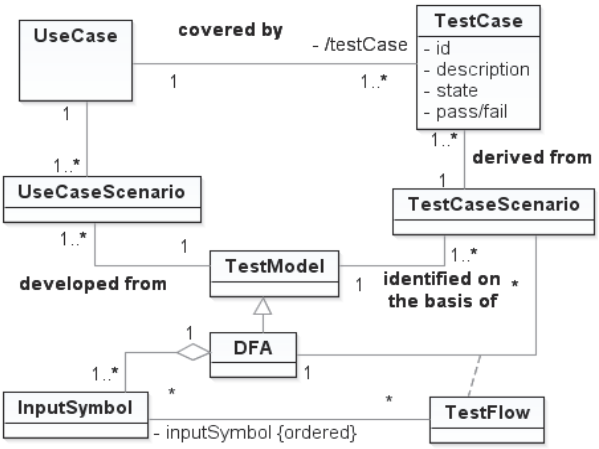

Figura 6. Empleo de AFD como modelo de prueba.

TestModel y *DFA*. El modelo de prueba es útil para definir criterios de cobertura sobre las pruebas a realizar y el porcentaje de cobertura que se desea cumplir con ese criterio sobre los casos de pruebas generados. Por ejemplo, considerando que se toman AFDs como modelos de prueba, un criterio podría ser la cantidad de estados probados / la cantidad total de estados del AFD. Otra opción, por ejemplo, es la cantidad de escenarios de excepciones probados / cantidad total de escenarios de excepciones.

Por ejemplo, para el caso de uso *CU-01* de la Figura 4 y el modelo de pruebas dado por el AFD de la Figura 5(a), es posible proponer los flujos de la Figura 5(b) con un criterio de cobertura de estados del 84% (16 estados probados / 19 estados totales). En la Figura 7 se presentan los flujos que se consideraron para la especificación de casos de pruebas, dados por las cadenas *NNN*, *NNANNNNANNNNAAN*, y *NAANNNNNNNN*, las cuales son secuencias válidas para el modelo de prueba. Estas secuencias son parte de la información del escenario de caso de prueba, como puede observarse en la asociación de clase (*TestFlow*) en el modelo conceptual de la Figura 6.

Una ventaja importante que se logra con este enfoque para derivar casos de pruebas a partir de casos de uso son los enlaces de trazabilidad que se establecen entre los casos de usos y los casos de prueba. Esta trazabilidad entre artefactos permite determinar si un caso de uso es cubierto por los casos de pruebas definidos en el grado que se desea.

Las herramientas de soporte para la definición de casos de uso no siempre están integradas con las herramientas con que se cuenta para la generación y ejecución de casos de prueba. Sin embargo, dado que los casos de prueba pueden ser derivados a partir de la especificación de casos de uso, y que los casos de prueba, a su vez, requieren de trazabilidad hacia los requerimientos funcionales y casos de uso, se requiere la integración o interoperabilidad entre dichas herramientas. Es necesaria la construcción de una ontología que defina todos los conceptos involucrados y las relaciones entre éstos, a fin de lograr tal interoperabilidad semántica. Tomando como base los modelos conceptuales definidos y siguiendo una metodología para el desarrollo de ontologías, se llevará a cabo la construcción de dicha ontología de requerimientos funcionales.

3. Construcción de la ontología

En los últimos años, el empleo de las ontologías en el contexto de desarrollo de software ha tenido un creciente protagonismo. Una ontología es una especificación de una conceptualización, donde tal conceptualización es entendida como una versión abstracta y simplificada del mundo que representa. Además, una ontología se refiere

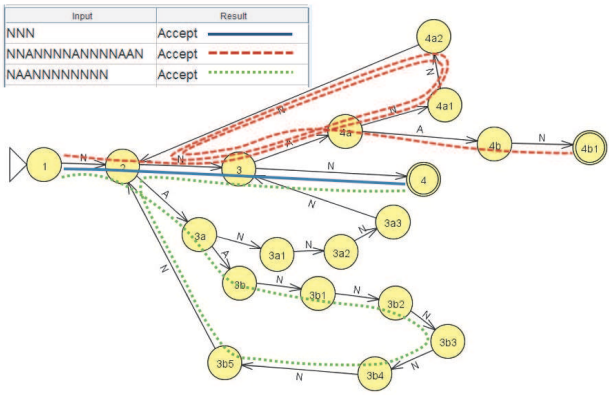

Figura 7. Empleo de AFD como modelo de prueba.

a un modelo de dominio formal que describe los conceptos y relaciones de ese dominio [19]. Las ontologías posibilitan la clasificación jerárquica de conceptos de dominio interrelacionados y pueden ser representadas empleando un esquema RDF o el lenguaje OWL. El uso de estos lenguajes permiten que las ontologías sean legibles por los humanos e interpretable por las máquinas, permitiendo realizar consultas para obtener conocimiento, así como también inferir nuevo conocimiento. Además, pueden aplicarse en la ingeniería de requerimientos para minimizar o resolver diferentes tipos de problemas. Por ejemplo, la obtención de mejoras en la especificación de requerimientos ya que contribuyen a la desambiguación de su significado, posibilitan el análisis de consistencia en requerimientos, permiten modelar en forma explícito el conocimiento del dominio, gestionar el conocimiento y el cambio de requerimientos, y lograr trazabilidad de requerimientos y otros artefactos [6]. Además, los lenguajes de la Web Semántica, tales como RDF y OWL facilitan la interoperabilidad en forma significativa. Proveen una estructura social y un marco de trabajo técnico para reusar ontologías existentes; y mecanismos formales para expresar equivalencia lógica entre clases y propiedades en diferentes ontologías.

Para desarrollar la ontología de requerimientos funcionales en primer lugar se determinó el dominio y alcance de la misma. Para ello se buscó responder a preguntas básicas del tipo: ¿Cuál es el dominio de aplicación de la ontología? ¿Cuál es la utilidad de la ontología? ¿Para qué tipos de preguntas el conocimiento contenido en la ontología debería brindar respuestas? ¿Quiénes usarán y mantendrán la ontología?

Una manera de determinar el alcance de la ontología es bosquejando una lista de preguntas de competencia, según la metodología de Gruninger y Fox [7]. Además, estas preguntas servirán posteriormente como una manera de validar o controlar la calidad de la ontología definida, ya que permitirán verificar si ésta contiene

suficiente información para responder a ese tipo de preguntas, y ver si las respuestas requieren un nivel particular de detalle o representación en un área en particular. Las preguntas de competencia son preliminares y no necesitan ser exhaustivas.

Siguiendo esta metodología, surgieron diferentes preguntas de competencia, las cuales fueron clasificadas en diferentes grupos para una mejor organización. Los grupos son: i) Definición de requerimientos funcionales, ii) Definición de casos de uso, iii) Definición de modelo de prueba, iv) Definición de casos de prueba, y v) Trazabilidad. En la Tabla 1 se explicitan las preguntas, categorizándolas dentro de las agrupaciones mencionadas.

A partir de las preguntas de competencia que se definieron, se identificaron y enumeraron los términos que son importantes para la ontología. Inicialmente, se generó una lista de términos sin tener en cuenta si existía solapamiento entre los conceptos que representan, las relaciones entre los términos, o las propiedades que caracterizan a dichos conceptos. Estos términos son los que se listan en la segunda columna de la Tabla 1.

El paso siguiente en la construcción de la ontología propuesta, fue, a partir de estos términos definir cada una de las clases que constituyen la jerarquía de clases de la ontología y sus propiedades [19]. Se empleó un proceso de desarrollo que utiliza una combinación de los enfoques top-down y bottom-up, comenzando por definir los conceptos más sobresalientes, luego generalizarlos y/o especializarlos apropiadamente.

Las clases por sí solas no proveen suficiente información para responder las preguntas de

Tabla 1. Preguntas de competencia.

Preguntas de Competencia	Términos identificados	Grupo
¿Cuáles son los requerimientos funcionales de un sistema? ¿Qué artefactos se emplean para su definición?	Requerimiento funcional, Sistema, Artefactos	Definición requerimientos funcionales
¿Cuáles son los casos de uso especificados para definir un requerimiento funcional dado?	Casos de uso, Requerimiento Funcional	Definición requerimientos funcionales
¿Cuál es el objetivo o meta de un caso de uso dado?	Objetivo, Caso de uso	Definición de caso de uso
¿Qué actores están involucrados en un caso de uso?	Actor, Caso de uso	Definición de caso de uso
¿Cuáles son las precondiciones de un caso de uso?	Precondiciones, Caso de Uso	Definición de caso de uso
¿Cuál es el escenario principal de un caso de uso? ¿Qué pasos abarca un escenario alternativo dado? ¿Existen alternativas o excepciones para el escenario principal?	Caso de Uso, Escenario principal, Paso de escenario	Definición de caso de uso
¿Cuáles son los escenarios alternativos de un caso de uso? ¿Qué pasos están involucrados en un escenario alternativo dado?	Caso de Uso, Escenario alternativo, Paso de escenario	Definición de caso de uso
¿Cuáles son los escenarios de excepción de caso de uso? ¿Qué pasos están involucrados en un escenario de excepción?	Caso de Uso, Escenario de excepción, Paso de escenario	Definición de caso de uso
¿Cuáles son las postcondiciones de un caso de uso dado?	Postcondiciones, Caso de Uso	Definición de caso de uso
¿Dado un requerimiento funcional expresado en un caso de uso, cuál es el autómata finito determinista que lo modela?	Requerimiento funcional, Caso de uso, Autómata finito determinista	Trazabilidad
¿Cuál es el alfabeto de símbolos de entrada manejada por cierto autómata finito determinista? ¿Cuáles son los estados? ¿Cuáles son las transiciones?	Alfabeto, Símbolos de entrada, Estados, Transiciones	Definición de modelo de comportamiento/prueba
¿Cuáles son las correspondencias o equivalencias de los estados de un autómata finito determinista y los pasos de los diferentes escenarios dentro de un caso de uso?	Estados, Pasos de escenario, Caso de uso	Definición de modelo de comportamiento/prueba
¿Dada una bifurcación en un curso de acción de caso de uso, cuáles son las transiciones correspondientes en la función de transición del autómata finito determinista? ¿Qué condiciones se verifican?	Transiciones, Condiciones	Definición de modelo de comportamiento/prueba
¿Cuáles son los posibles estados finales alcanzados por el autómata finito determinista? ¿Cuáles de esos estados finales se producen por condiciones de error?	Estado, Estado final, Estado final por error	Definición de modelo de comportamiento/prueba
¿Qué tipo de modelos se utiliza para transformar especificaciones de comportamiento de requerimientos funcionales a modelos de prueba?	Modelo de prueba	Definición de modelo de comportamiento/prueba
¿Cuáles son los escenarios de casos de prueba definidos para un cierto caso de uso?	Escenario de caso de prueba, Caso de uso	Definición de casos de prueba
¿Qué casos de prueba se derivaron a partir de un caso de uso?	Caso de prueba, Caso de uso	Trazabilidad
¿Cuáles son los casos de prueba que se instancian a partir de cierto escenario de caso de prueba?	Caso de prueba, Escenario de caso de prueba	Definición de casos de prueba
¿Cuáles son los tipos de datos de entrada que se definen para un cierto escenario de caso de prueba?	Tipo de dato de entrada, Escenario de caso de prueba	Definición de casos de prueba
¿Cuáles son los tipos de resultados esperados que de definen para un cierto escenario de caso de prueba?	Tipo de resultado esperado, Escenario de caso de prueba	Definición de casos de prueba
¿Qué criterio de cobertura se definió para la definición de los escenarios de caso de prueba?	Escenario de caso de prueba, Criterio de cobertura	Definición de casos de prueba
¿Qué porcentaje de cobertura se aplicó para la definición de escenarios de casos de prueba?	Escenario de caso de prueba, Porcentaje de cobertura	Definición de casos de prueba
¿Qué flujos de pruebas o secuencias de estados se seleccionaron para la derivación de casos de prueba a partir de un cierto modelo de prueba?	Flujo de prueba, Secuencia de estados, Modelo de prueba	Definición de casos de prueba

competencia, por lo que es necesario describir la estructura interna de los conceptos que representan. Para ello, a partir de los términos identificados se analizó si éstos constituían propiedades intrínsecas, extrínsecas, partes (si el objeto que se está describiendo es estructurado) o relaciones con otros conceptos.

Posteriormente, la construcción de la ontología se llevó a cabo empleando la herramienta Protégé. Protégé incluye clasificadores deductivos para validar que los modelos sean consistentes y para inferir nueva información en base al análisis de una ontología. El modelo creado se representó en lenguaje OWL el cual es un lenguaje de etiquetado semántico para publicar y compartir ontologías en la Web. Se trata de una recomendación del W3C, y puede emplearse para representar ontologías de forma explícita, es decir, permite definir el significado de términos en vocabularios y las relaciones entre aquellos términos. En realidad, OWL es una extensión del lenguaje RDF (Resource Description Framework) y emplea las tripletas de RDF, aunque es un lenguaje con más poder expresivo que éste.

El uso de este lenguaje tiene la ventaja de que está diseñado para usarse cuando la información contenida en los documentos necesita ser procesada por programas o aplicaciones, en oposición a situaciones donde el contenido solamente necesita ser presentado a los seres humanos, lo cual lo hace adecuado para los objetivos de recuperación y reuso de conocimiento que tiene el enfoque del que forma parte la ontología.

De esta manera, utilizando OWL, la ontología se formalizó en términos de clases, individuos, y propiedades de estos individuos y clases, así como relaciones que existen entre ellas. En particular, OWL permite expresar los siguientes diferentes tipos de propiedades: (i) propiedades de objeto, las cuales permiten definir relaciones entre individuos; (ii) propiedades de tipo de dato, las cuales definen relaciones entre individuos y literales (por ejemplo, strings, enteros, etc.); y propiedades de anotaciones (annotations), que pueden ser usadas para describir metadatos de individuos, clases y propiedades, tales como el idioma en que se encuentra, definiciones y comentarios de estos conceptos. Además, sobre las propiedades de objeto definidos se especificó el dominio y rango, así como también tipo, valores admitidos y cardinalidad, sobre las propiedades de dato. En la Figura 8 se presenta una vista parcial de la especificación de la ontología en Protégé, donde se incluyen los conceptos que fueron mencionados.

Para completar la definición de la ontología, y dada la suposición de mundo abierto que se tiene en Protégé, se agregaron los axiomas de disyunción, cierre, y cobertura necesarios de manera que sea posible ejecutar razonadores sobre la ontología. Dado que OWL asume que las clases se solapan entre ellas a menos que explícitamente se diga lo contrario, se incorporaron axiomas de disyunción indicando qué clases son disjuntas.

OWL adhiere al principio de no unicidad de nombres. Este principio sostiene que por tener diferentes nombres dos recursos no necesariamente son individuos distintos, podrían ser el mismo recurso con denominaciones diversas. Por lo tanto, en el dominio cerrado en el que se está proponiendo la ontología, al crear las instancias se debe especificar que se trata de distintos individuos. Finalmente, se definieron un conjunto de reglas que permiten la inferencia de conocimiento y la especificación de restricciones que no estaban explícitas

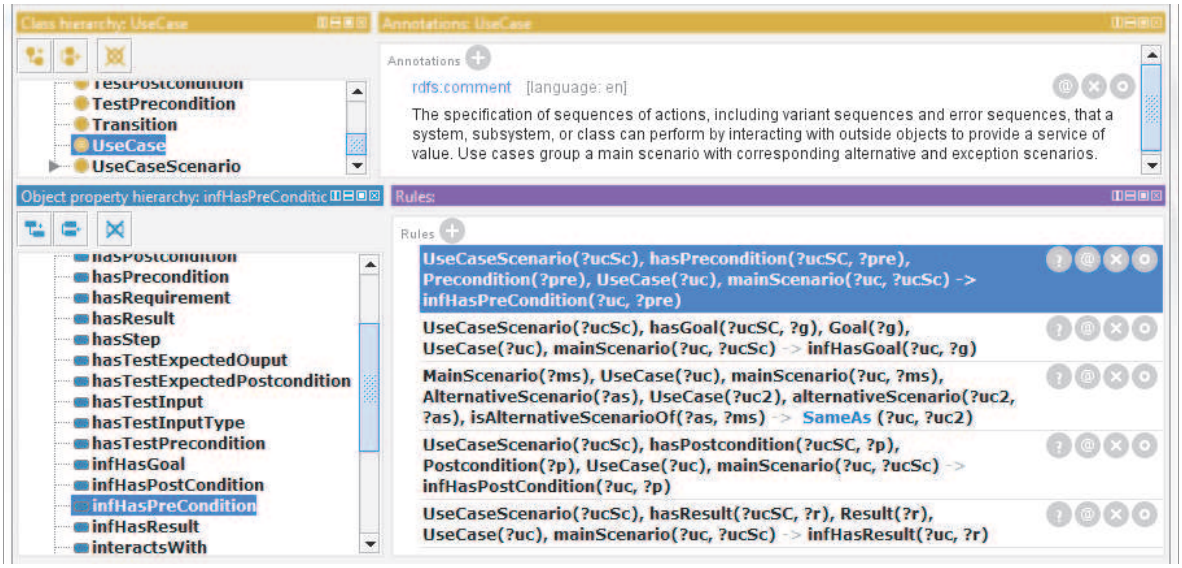

Figura 8. Vista parcial de la especificación de la ontología en Protégé.

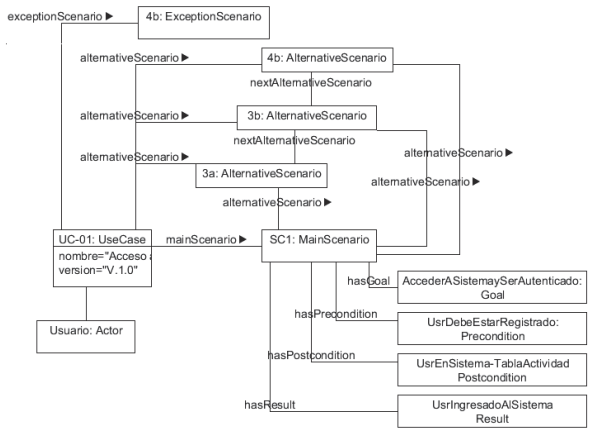

Figura 9. Modelo de objetos que representan el caso de uso CU-01.

entre las propiedades de objeto definidas. En los párrafos siguientes se explicará brevemente la definición de algunas de estas reglas por medio de un ejemplo.

La Figura 9 presenta un modelo de objetos que representa el caso de uso *CU-01* que fuera introducido en la Figura 4. Las instancias que se muestran en el mencionado modelo se definieron como individuos (*owl:NamedIndividual*) en la ontología. En la parte inferior derecha de la Figura 8, se muestran algunas de las reglas que se definieron en la ontología. En particular se muestran las reglas que permiten inferir las relaciones *infHasGoal*, *infHasPreCondition*, *infHasPostCondition* e *infHasResult* que vinculan la clase *UseCase* con las clases *Goal*, *Precondition*, *Postcondition* y *Result*, respectivamente. En todos los casos, se obtienen las correspondientes relaciones navegando por las asociaciones explícitas que vinculan un caso de uso con sus escenarios y dichos escenarios con sus objetivos, precondiciones, postcondiciones y resultados. En la Figura 10 se ilustran estas inferencias.

Además de las reglas que permiten la inferencia de conocimiento nuevo, otro conjunto de reglas especifican restricciones de consistencia de los modelos. Por ejemplo, es posible definir reglas para indicar, si un escenario principal está asociado a determinados escenarios alternativos, todos ellos son escenarios del mismo caso de uso. De forma similar, si un paso de escenario es el paso siguiente a otro paso de escenario, ambos pasos pertenecen a escenarios del mismo caso de uso. La definición de la primera de las reglas mencionadas puede observarse en la Figura 8. En particular, dicha regla especifica que si un escenario principal *?ms*, asociado a un caso de uso *?uc*, tiene un escenario alternativo *?as* y, al mismo tiempo, este escenario alternativo está asociado a otro caso de uso *?uc2*, entonces *?uc* y *?uc2* son el mismo individuo. Pero dado, que en la creación y población de la ontología se especificó que todos los individuos son diferentes entre sí, si se llega a dar esta situación la ontología quedaría inconsistente. A fin de mostrar cómo esta regla permite detectar las violaciones de las restricciones representadas, se agregó a la ontología una instancia de *AlternativeScenario* (denominada *3adeOtroCaso* en la Figura 11), que se vincula con el escenario principal ya definido (*SC1* en Figura 9) y con un nuevo caso de uso (*CU-02* en Figura11). Esta situación violaría la restricción indicada en los párrafos previos (un escenario principal debe estar asociado al mismo caso de uso al cual están asociados sus escenarios alternativos). Una vez actualizado el ejemplo con los nuevos objetos, se sincroniza el razonador, el cual detecta que la ontología queda inconsistente luego de la modificación y presenta los motivos (Figura 11).

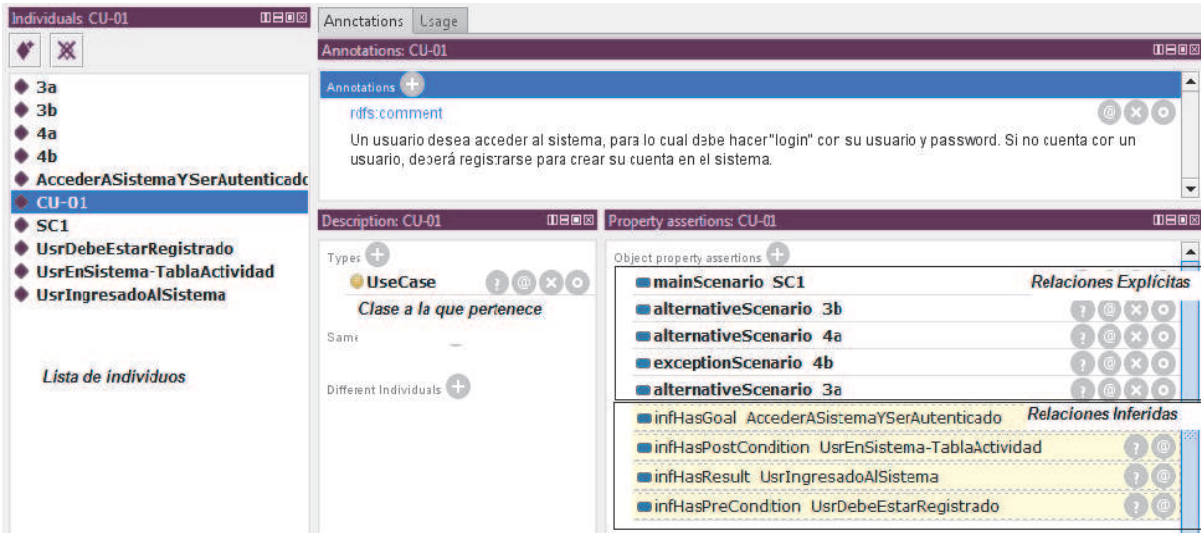

Figura 10. Inferencias para el caso de uso CU-01.

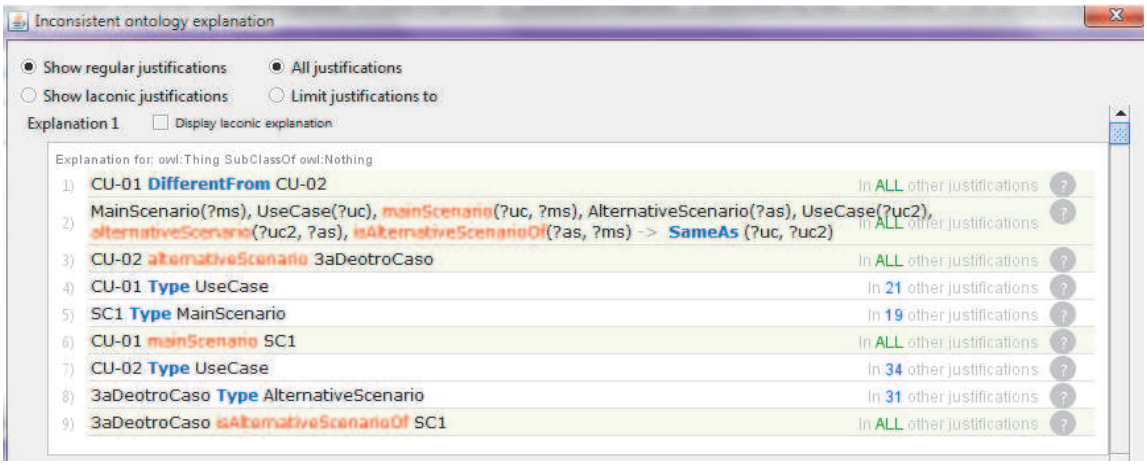

Figura 11. Inconsistencia detectada por un razonador en Prótegé

4. Trabajos relacionados

Existen diversos trabajos que se basan en ontologías para la mejora de las actividades de Ingeniería de Requerimientos, pero pocos han aplicado éstas para lograr interoperabilidad entre herramientas para la especificación de casos de uso y herramientas de gestión de casos prueba. Una revisión sistemática de literatura en el uso de ontologías en ingeniería de requerimientos, que abarca 66 publicaciones entre 2007 y 2013 [6], reporta que sólo algunas de las ontologías desarrolladas abarcan más de una fase del proceso de ingeniería de requerimientos. La gran mayoría se limitan a la fase de especificación, seguidas por algunas que incluyen también análisis, gestión y elicitación. Finalmente una pequeña minoría menor al 7% de los trabajos consideran las etapas de validación. Dentro de las pocas propuestas de tratan la temática de validación de requerimientos, se encuentra una basada en modelos de comportamiento sobre requerimientos [13], que fundamentalmente está enfocada en sistemas distribuidos que carecen de un control central. En ella se utilizan "Message Sequence Charts" (MSC), lenguaje estandarizado por la ITU (Union Internacional de Telecomununicaciones).

En relación a la generación de casos de prueba a partir de escenarios de casos de uso, Pohl y colab. [16, 17] propusieron una estrategia basada en el uso de modelos de prueba. Como modelos de prueba utilizan diagramas de actividades UML, y definen una técnica denominada SCenTed la cual fue concebida para líneas de producto de software.

La propuesta de Kof y colab. [9] aborda la aplicación de ingeniería ontológica en la ingeniería de requerimientos, específicamente durante la transición de requerimientos funcionales expresados en lenguaje natural a la fase de diseño de un sistema. Los autores afirman que el principal desafío radica en que los documentos de requerimientos son informales (tienen inconsistencias y omisiones), mientras que el código que se debe construir es formal, por lo que los modelos de sistemas, como un paso intermedio entre los requerimientos y el código, contribuyen a la comprensión de los requerimientos. Tomando esta premisa, los autores extraen un modelo conceptual (una ontología) y un modelo de comportamiento a partir de diferentes fuentes. Luego, comparan estos modelos y validan su consistencia. A diferencia de nuestra propuesta emplean los modelos intermedios para validación entre las fases de requerimientos y diseño de un sistema, mientras que la nuestra se enfoca en emplearlos como elemento de validación interna de la fase requerimientos y con la fase de pruebas.

Otro trabajo relacionado a la especificación de casos de uso mediante ontologías es el de Bagiampou y Kameas [2], en el cual se propone una ontología cuyos conceptos surgen de la notación de diagramas de casos de uso UML. Esta ontología puede ser usada por diseñadores de aplicaciones de e-learning, y su objetivo no está enfocado en la interoperabilidad de herramientas sino en educación en ingeniería de software, como recurso de aprendizaje. La ontología puede ser empleada para describir diagramas de casos de uso y como repositorio de conocimiento para los usuarios (estudiantes) a fin realizar consultas generales sobre el dominio de diagramas de casos de uso o consultas sobre términos y sus relaciones en un diagrama de caso de uso específico.

5. Conclusiones y trabajos futuros

En este trabajo se presentó un enfoque basado en ontologías para la especificación, validación y trazabilidad de requerimientos funcionales basados en casos de uso estructurados y casos de prueba. La propuesta conlleva las siguientes ventajas:

i) Posibilita que las características del sistema que son

relevantes para los stakeholders clientes o usuarios sean documentadas como artefactos de tipo plantilla de casos de uso;

ii) Propone una técnica para la generación de un modelo de caso de uso basado en AFD que posibilita la validación de los casos de uso;

iii) Utiliza modelos de pruebas basados en AFD, lo que permite derivar un conjunto de casos de pruebas, especificando cierto criterio de cobertura. Esta técnica además, posibilita detectar fallas o defectos en requerimientos;

iv) La ontología propuesta propicia la interoperabilidad entre herramientas de soporte HECU y HGCP, ya que se basa en un modelo conceptual que soporta los metamodelos de esas herramientas.

Se espera extender la ontología propuesta, incorporando conceptos para la representación de requerimientos de calidad, tomando como base una ontología preliminar para la documentación de un esquema de calidad basada en el estándar ISO/IEC 25010 [3]. Otra posibilidad de extensión de la presente ontología es incorporar los conceptos y reglas relativas a la priorización de nuevos requerimientos [5].

6. Agradecimientos

Este trabajo ha sido financiado por la Universidad Tecnológica Nacional (Proyecto IPN4409 "Herramientas y Métodos de soporte a la Ingeniería de Software: requerimientos, estrategias ágiles, y calidad de procesos y productos") y el CONICET. Se agradece el apoyo brindado por estas instituciones.

7. Referencias

[1] Allmann, C., Denger, C., and Olsson, T, *Analysis of Requirements-based Test Case Creation Techniques*, IESE-Report No. 046.05/E, 2005.

[2] Bagiampou, M., Kameas, A., "A Use Case Diagrams ontology that can be used as common reference for Software Engineering education", *2012 6th IEEE International Conference Intelligent Systems*, Sofia, 2012, pp. 035-040.

[3] Blas, M. J. and Gonnet, S., "An Ontological Approach to Analyze the Data Required by a System Quality Scheme", 44 JAIIO 1er SAOA, Rosario, 2015, pp. 111-120.

[4] Cockburn, A., *Writing Effective Use Cases (1st ed.)*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2000.

[5] De Federico, S. and Gonnet, S. , "New requirements prioritization based on customer historical profiles," 2016 XLII Latin American Computing Conference (CLEI), Valparaiso, 2016, pp. 1-8.

[6] Dermeval, D. , Vilela, J., Bittencourt, I. I., Castro, J., Isotani, S., Brito, P. , "A Systematic Review on the Use of Ontologies in Requirements Engineering", *2014 Brazilian Symposium on Software Engineering*, Maceio, 2014, pp. 1-10.

[7] Gruninger M., Fox M. S., "Methodology for the Design and Evaluation of Ontologies", *IJCAI Workshop on Basic Ontological in Knowledge Sharing*, Montreal, Canada, 1995.

[8] Hopcroft, J., Motwani, R., Ullman, J., *Introducción a la teoría de autómatas, lenguajes y computación*, 3 ed., Pearson, 2007.

[9] Kof, L., Gacitua, R., Rouncefield, M., Sawyer, P. , "Ontology and Model Alignment as a Means for Requirements Validation", *IEEE Fourth International Conference on Semantic Computing (ICSC)*, 2010, pp. 46–51.

[10] Marciszack, M., Perez Cota, M., and Groppo, M., "Metodología y Herramienta de soporte para validar Modelos Conceptuales a través de Máquinas Abstractas", Revista de Ciencia y Tecnología, U. de Palermo, Nro 15, págs. 165-180.

[11] Medina, O., Marciszack, M., Groppo, M. "Herramienta para administración y validación de requerimientos de sistemas". 4th CIMPS 2015, "Tendencias en la Ingeniería del Software. Impacto en las Tecnologías de Información y Comunicación", Editorial: CIMAT A.C., México, 2015.

[12] Medina, O., Marciszack, M., Groppo, M., "Trazabilidad y validación de requerimientos funcionales de sistemas informáticos mediante la transformación de modelos conceptuales". Revista ReCIBE, Año5 No. 1, Universidad de Guadalajara. Guadalajara, Jalisco, México, 2016.

[13] Moshirpour, M., Mireslami, S., Alhajj, R., Far, B. H., "Automated ontology construction from scenario based software requirements using clustering techniques", *IEEE 13th International Conference on Information Reuse & Integration (IRI)*, Las Vegas, NV, 2012, pp. 541-547.

[14] Pohl, K., *Requirements Engineering: Fundamentals, Principles, and Techniques*, Springer, 2010.

[15] Pretschner, A., Prenninger, W., Wagner, S., Kühnel, C., Baumgartner, M., Sostawa, B., Zölch, R., Stauner, T., "One evaluation of model-based testing and its automation." *Proceedings of the 27th international conference on Software engineering (ICSE '05)*, ACM, NY, USA, 2005, pp. 392-401.

[16] Reuys, A., Kamsties, E., Pohl, K. Reis, S. , "Model-Based System Testing of Software Product Families", *Advanced Information Systems Engineering, CAiSE* , Porto, Portugal, 2005, Springer Berlin Heidelberg, pp. 519-534.

[17] Reuys, A., Reis, S., Kamsties, E., Pohl, K., "The ScenTED Method for Testing Software Product Lines," *Software Product Lines*, 2006, Springer Berlin Heidelberg, pp. 479-520.

[18] Sommerville, I., *Software Engineering. 9th Edition*, Addison-Wesley, 2011.

[19] Uschold, M., Gruninger M., "Ontologies: Principles, Methods and Applications", *Knowledge Engineering Review*, 1996.

[20] Wiegers, K.E., *Software Requirements. 3rd Edition*, Microsoft Press, 2013.