UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

PROYECTO DE SISTEMAS INFORMÁTICOS
CURSO 2006-2007

Generación de cuentos interactivos usando CBR

Componentes del grupo:

Marina Gallego Barrigón
Irene Pérez Medina
Almudena Ruiz Iniesta

Profesora:

Mª Belén Díaz Agudo

Los abajo firmantes autorizan a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado, relativo al presente proyecto de Sistemas Informáticos.

_____	_____	_____
Marina Gallego Barrigón	Irene Pérez Medina	Almudena Ruiz Iniesta

Madrid, ___ de _____ de 2.007

Palabras clave

Generación de historias interactivas, Ontologías, Razonamiento basado en casos, Adaptación Constructiva, Sistemas inteligentes, Inteligencia Artificial.

Resumen

El proyecto consiste en un sistema interactivo que, a partir de una consulta que recibe del exterior y una base de cuentos genera nuevos cuentos haciendo modificaciones en los primeros.

La base de conocimiento se compone de cuentos populares infantiles que sirven de base para crear los nuevos. Estos cuentos son representados mediante una ontología que incluye una jerarquía basada en la definición realizada por Vladimir Propp en base al estudio que desarrolló sobre la estructura común de los cuentos populares rusos.

Como todo sistema interactivo una parte fundamental de este trabajo es la dedicada a la comunicación con el futuro usuario. Este flujo de información se lleva a cabo de tres formas diferentes. Una de ellas es a través del editor mediante el cual el usuario puede representar cuentos que se añaden a la base de conocimiento. No es necesario que el usuario tenga conocimientos de la representación ontológica debido a que la comunicación se desarrolla a través de una interfaz gráfica. El sistema traduce los datos introducidos por el usuario a su representación en la ontología. El segundo flujo de información corresponde con el generador de cuentos. El usuario elije una serie de parámetros que el generador tendrá en cuenta a la hora de recuperar información de la base de conocimiento para generar un nuevo cuento. Y el tercer y último flujo de información se desarrolla a lo largo del proceso de creación del nuevo cuento. Los cuentos se generan de manera gradual, lo que significa que en determinados momentos del proceso el generador puede ofrecer al usuario la posibilidad de decidir cómo continuar el cuento.

The project consists of an interactive system that, from the consult that receives from the outside and a tale's base generates new tales doing modifications on the firsts.

The knowledge base is made up of stories within the domain of fairytales that are used to generated new tales. These stories are represented by means of an ontology that include a hierarchy based on the definition made by Vladimir Propp on the basis of the study that he developed on the common structure of Russian popular stories.

As every interactive system a fundamental part of this work is that one dedicated to the communication with the future user. This flow of information is carried out of three different forms. One of them is through publisher used by the user to represent stories that are added to the knowledge base. It is not necessary that the user has knowledge of the ontological representation because the communication is developed through a graphical interface. The system translates the data introduced by the user to its representation in the ontology. The second flow of information corresponds with the story generator. The user chooses several parameters that the generator will consider at the time of recovering information of the knowledge base to generate a new story. And the third and last flow of information is developed throughout the process of creation of the new story. The stories are generated incrementally, which means that at certain moments of the process the generator can offer to the user the possibility of deciding how to continue the story.

ÍNDICE

1. INTRODUCCIÓN Y OBJETIVOS DEL PROYECTO

El procesamiento del lenguaje natural es una de las áreas de aplicación clásica de la Inteligencia Artificial. En este campo intervienen aspectos tan importantes y cuestionados como el de la creatividad de los programas informáticos. Partiendo de unos datos iniciales, ¿puede un programa ser creativo y transformar los datos de entrada de forma que se componga un texto original? por ejemplo, una historia literaria interesante, una poesía... Son muchos los dominios en los que se requiere la generación automática de textos. Por ejemplo, para generar cualquier tipo de crónica deportiva, como la de un partido de fútbol, o para la composición de un cuento o historia sobre cualquier tema (magia, misterio, terror, amor, desamor, peleas,...)

En este proyecto el dominio elegido es el de los cuentos populares infantiles, a partir de un conjunto de datos objetivos iniciales, se realizará una generación automática de un guión para el cuento del que después se podrá generar un texto que lo describa. En lo relativo a la generación del guión de la historia, la idea es hacer un guión no-lineal, inspirado en los famosos cuentos de *Elige tu propia Aventura*[1] , también conocidos como libros de juegos. Cuentos en los que se presenta el comienzo de la historia al lector quien llegado a un punto deberá tomar decisiones sobre el transcurrir de la historia. De esta forma la generación final del cuento dependerá de las decisiones que el usuario del sistema haya tomado durante la creación del guión.

El conocimiento del sistema es de distintos tipos: conocimiento del dominio (elementos que aparecen en el texto como los distintos tipos de personajes), lingüísticos (elementos de la lengua, la morfología, la sintaxis, la coordinación de números, géneros, personas y conjugaciones de verbos) y otros elementos de más alto nivel conceptual que coordinen la estructura del discurso. En nuestro caso limitaremos el conocimiento del sistema al conocimiento del dominio del mismo, teniendo en cuenta qué elementos pueden formar parte de un cuento, es decir, el conocimiento del sistema sólo será en base al dominio de los cuentos infantiles y de las partes que componen la representación de nuestros cuentos: acciones, personajes, lugares y objetos.

Durante el presente capítulo nos disponemos a explicar los objetivos del proyecto así como algunos conceptos necesarios para la mejor comprensión del trabajo.

[1] http://es.wikipedia.org/wiki/Elige_tu_propia_aventura

1.1 Objetivos

Al iniciar el proyecto nos marcamos unos objetivos muy generales, que poco a poco se fueron concretando. En la medida en que fue pasando el tiempo pudimos descubrir nuevos objetivos y objetivos poco viables. Para empezar, el objetivo principal fue que nuestro sistema sería un generador de narración interactiva, que tendría como dominio los cuentos populares infantiles. De aquí obteníamos dos objetivos. El primero, conseguir un buen generador de cuentos que fuera creativo y coherente. Como segundo, introducir interactividad a nuestro generador, para que el futuro usuario que utilice el sistema se sienta creador de su propio cuento. Como podemos ver, estos dos son objetivos muy generales que intentaremos refinar un poco más.

Para poder generar un cuento necesitamos tener representados de alguna manera nuestros cuentos en forma de casos. El primer objetivo fue cómo representaríamos esos casos. Finalmente se decidió que lo más sensato para facilitar el posterior razonamiento era construir una ontología donde representar todo el conocimiento necesario en el dominio de los cuentos infantiles. Esta ontología se creó usando como base *KIIDSOnto*[2]. Ahora bien, no es suficiente saber cómo vamos a representar los casos es muy importante saber de dónde vamos a obtener esos casos, es decir cómo realizaremos la obtención de casos de nuestro sistema. La adquisición de casos supuso un gran cuello de botella en nuestro sistema. Para poder generar cuentos originales, creativos y coherentes era muy necesario que la base de casos fuera rica y abundante, pero conseguir estas dos características sería muy costoso. Por eso, nos planteamos la posibilidad de crear un editor de cuentos para poder facilitar la adquisición de casos. La idea era crear una interfaz amigable para poder introducir datos en nuestra base de conocimiento de manera fácil y rápida.

Una vez que ya sabíamos cómo representaríamos los casos, sabiendo cómo haríamos la adquisición de los mismos, los siguientes objetivos giraban en torno a la generación interactiva de los cuentos, es decir, cómo sería nuestro sistema de razonamiento sobre los casos. Uno de los objetivos principales de este sistema de razonamiento era poder introducir interactividad en el proceso de generación de la historia. Para ello estudiamos dos formas de implementar el razonador de nuestro sistema. La primera de ellas sería un razonamiento basado en casos tradicional, que tras recuperar un caso de la base de conocimiento lo adaptaría si fuese necesario para mostrarlo al usuario. Sin embargo, de este modo era difícil incluir la interactividad ya que los casos se recuperan enteros y no habría manera de combinarlos. La segunda opción era hacer un razonamiento basado en casos generativo, de manera que fuese construyendo el cuento poco a poco, pudiendo combinar elementos de diferentes casos de la base de conocimiento. De esta manera, incluir la interactividad era mucho más sencillo ya que el cuento se va generando poco a poco y se pueden determinar puntos en los que no sea el sistema quien elija cual es el siguiente paso de la construcción si no que sea el usuario.

[2] KIIDSOnto es la ontología de KIIDS [14] trabajo que explicaremos en la siguiente sección.

Por último como objetivo final, pero que será una parte de trabajo futuro, nos planteamos la generación de lenguaje natural para los cuentos obtenidos. Como solución al problema se obtendría un guión compuesto por los elementos que conforman el cuento: las acciones, los personajes, los lugares, los objetos, y en una última etapa esto se pasaría a lenguaje natural. Sin embargo, esto es bastante complicado ya que hay que tener en cuenta muchas cosas como el género y número de las palabras, de los verbos, la concordancia de tiempos... En nuestro trabajo hemos optado por el uso de plantillas para representar el cuento obtenido en lenguaje natural dejando como trabajo futuro profundizar en esta parte.

Podemos decir que los objetivos del proyecto se resumen, en nuestro caso, a tres:

- Base de conocimiento sólida y estable, rica en conceptos así como en las propiedades necesarias para establecer relaciones con cada uno de los elementos constitutivos de una historia.
- Editor de cuentos con una interfaz sencilla y amigable, capaz de enriquecer nuestra base de conocimiento con nuevos casos.
- Sistema generador del guión de cada historia en el que sea posible la participación por parte del usuario en la generación del guión.

1.2 Ontologías

El primer problema con el que nos encontramos, y que es una de las principales partes en las que se puede dividir nuestro proyecto, era representar el conocimiento sobre el dominio de los cuentos populares infantiles con el fin de poder posteriormente razonar sobre ello y generar cuentos.

La formalización del conocimiento podía hacerse de diferentes maneras. Teníamos la posibilidad de representar el conocimiento a través de reglas que formalizaran todo lo necesario para representar los cuentos. Sin embargo, esta opción era demasiado complicada ya que los elementos que componen un cuento son muchos y variados lo que provocaría la necesidad de crear un gran número de reglas dificultando también el razonamiento posterior sobre dichas reglas. Además, cada vez que se añadiese un cuento nuevo se necesitará añadir nuevas reglas lo cual posiblemente no fuera algo sencillo ni que se pudiese automatizar fácilmente puesto que estamos tratando un dominio extenso y complejo.

La otra opción, la que decidimos utilizar para el sistema era representar el conocimiento creando una ontología donde se definiesen todos los elementos necesarios y que simplificase sustancialmente tanto la representación como el razonamiento.

En este apartado explicaremos qué son las ontologías.

¿Qué es una Ontología?

Ontología[3] en Filosofía significa "*una explicación sistemática de lo que existe*". Es una rama de la metafísica que trata de describir o proponer las categorías y relaciones básicas del ser o la existencia para definir las entidades y decir de qué tipo son**.**

En Inteligencia Artificial, y en particular en Ingeniería del Conocimiento, la palabra ontología se utiliza para denotar una base de conocimiento reutilizable, es decir, una representación explícita, expresada en un lenguaje formal, del conocimiento acerca de un dominio.

> *"Ontology is a hierarchically structured set of terms for describing a domain that can be used as a skeletal foundation for a knowledge base"* [32]
> *"Ontology provides the means for describing explicititly the conceptualization behind the knowledge represented in a knowledge base"*. [33]

Al contrario de lo que ocurre en Filosofía, en los sistemas basados en el conocimiento, lo que existe es exactamente lo que se puede representar y formalizar. Una Ontología, en este sentido, hace referencia a un intento de formular un esquema conceptual dentro de un dominio dado de manera que pueda ser reutilizable por cualquier otro sistema que trabaje sobre el mismo dominio. Una de las definiciones más famosas de lo que es una ontología y que hace referencia al concepto de conceptualización es la dada por Gruber en 1993:

> *"a formal, explicit specification of a shared conceptualization"* [30]

El termino conceptualización es uno de los más usados como sinónimo de ontología y se refiere a un modelo abstracto de algún fenómeno del mundo del que se identifican los conceptos que son relevantes. Pero una ontología es mucho más que una taxonomía de conceptos. Las ontologías se componen también de instancias, relaciones que representan un tipo de interacción entre conceptos del dominio, funciones que son consideradas como un tipo especial de relación y axiomas que son proposiciones siempre verdaderas y que se expresan en un lenguaje lógico.

Se pueden distinguir tres tipos fundamentales de ontologías[4]:

- **Ontologías de un dominio**: en este tipo de ontologías se representa conocimiento especializado sobre un dominio o subdominio dado. En ellas se representan todos los conceptos, relaciones que rigen el comportamiento en ese dominio. Este tipo de ontologías se podrían usar en cualquier aplicación que tratase sobre el mismo dominio que conceptualiza la ontología. Por ejemplo, una ontología que representase conocimiento sobre la medicina o sobre las aplicaciones militares sería una ontología de este tipo.

[3] En estas direcciones podemos encontrar más información sobre qué es una ontología.
http://www.w3.org/TR/owl-guide/
http://www-ksl.stanford.edu/kst/what-is-an-ontology.html

[4] Según la clasificación propuesta por Steve et al. (1998)

- **Ontologías genéricas**: son aquellas en las que se trata se representar conceptos generales y fundacionales del conocimiento como las estructuras parte/todo, la cuantificación, los procesos.

- **Ontologías representacionales o meta-ontologías** *(top-level ontologies*): este tipo de ontologías proporcionan elementos y términos con un alto nivel de abstracción, es decir, lo que conceptualizan son términos generales bajo los cuales se suelen colocar los términos más específicos de otras ontologías.

Existen otros tipos de ontologías pero estos son los principales. Además, una misma ontología podría tener características de varios tipos diferentes.

¿Cómo construir una ontología?

Para la construcción de ontologías existen varias metodologías pero ninguna de ellas está estandarizada y cada equipo de desarrollo puede utilizar diferentes criterios o tomar decisiones de diseño de manera personal sin seguir ningún patrón. Es un trabajo bastante subjetivo al diseñador.

Las metodologías existentes[5] coinciden en comenzar la construcción de una ontología por determinar correctamente el dominio sobre el que vamos a hacer la ontología, identificando los procesos y adquiriendo conocimiento sobre el dominio que se desea representar. Tras esto las diferentes metodologías difieren en la manera de continuar la construcción de la ontología.

Una manera de crear la ontología, tras la fase de adquisición de conocimiento, es determinar las clases que forman el dominio, organizarlas en una jerarquía taxonómica, definir las propiedades de cada clase indicando las restricciones de sus valores y en el caso de querer crear instancias habrá que asignar valores a las propiedades.

Para la formalización de las ontologías existen diferentes lenguajes. Algunos de ellos están basados en alguno de los paradigmas de representación del conocimiento clásicos como son EXPRESS [28], CML [29], Ontolingua [30], entre otros.

De un tiempo a esta parte, debido al intercambio de ontologías a través de la Web se han desarrollado nuevos lenguajes para la especificación de ontologías. Estos nuevos lenguajes se basan en los estándares Web, entre los que cabe destacar RDF (*Resource Description Framework* [24]) y OWL (*Ontology Web Language* [17]).

RDF se basa en la idea de convertir las declaraciones de los recursos en expresiones con la forma sujeto-predicado-objeto (conocidas en términos RDF como tripletes). El sujeto es el recurso, es decir aquello que se está describiendo. El predicado es la propiedad o relación que se desea establecer acerca del recurso. Por último, el objeto es el valor de la propiedad o el otro recurso con el que se establece la relación.

[5] metodología de Uschold (1996), la de Grüninger y Fox (1995) o METHONTOLOGY (1999) son algunas de ellas

Otro de los lenguajes usados para la especificación de ontologías es OWL. Se trata de un lenguaje de marcado para publicar y compartir datos usando ontologías en la Web. OWL tiene como objetivo facilitar un modelo de marcado construido sobre RDF.

```
<owl:Class rdf:ID="Place">
   <owl:disjointWith>
     <owl:Class rdf:ID="Entity"/>
   </owl:disjointWith>
   <rdfs:subClassOf>
     <owl:Class rdf:ID="SOntoThing"/>
   </rdfs:subClassOf>
   <rdfs:comment rdf:datatype="http://www.w3.org/2001/XMLSchema#string"
   >modelado de los lugares...</rdfs:comment>
  </owl:Class>
```

Figura 1.1: Ejemplo de definición de una clase en OWL

Ejemplos de Ontologías

Existen varios proyectos que desarrollan ontologías y que tienen gran importancia. Por ejemplo,

- Cyc [26]: es una ontología de tipo meta-ontología. En ella se trata de representar conocimiento general con el fin de permitir a los sistemas que lo usen realizar razonamientos del tipo humano haciendo inferencias sobre el conocimiento que tiene representado.

- Mikrokosmos [27]: este es un proyecto, cuya ontología es su componente central, que ha sido desarrollado por el Computing Research Laboratory (CRL) de la New Mexico State University en Estados Unidos. Se trata de un sistema práctico a gran escala, enfocado en principio a traducir entre los idiomas inglés y español, y que actualmente está siendo expandido para dar cabida a otros idiomas.

1.3 Razonamiento basado en casos

El siguiente paso de nuestro proyecto era definir e implementar el modo en el que se haría el razonamiento del sistema para generar los cuentos. Para ello estudiamos dos opciones posibles que van a ser explicadas en este apartado. Por un lado teníamos la posibilidad de implementar un razonador basado en casos de manera tradicional como veremos en el apartado 1.3.1 y por otro lado, la adaptación constructiva que es otro tipo de razonamiento también basado en casos pero que funciona de otra manera como explicaremos más adelante en el apartado 1.3.2.

1.3.1 CBR transformacional

El razonamiento basado en casos (CBR, del inglés *Case Based Reasoning)* se basa en el modo en que los humanos razonamos para solucionar problemas cotidianos usando la experiencia previa que tendemos de situaciones parecidas. De esta manera, cuando nos encontramos ante un problema podemos recordar alguna situación similar y comportándonos como ya lo hicimos en el pasado, adaptando si fuera necesaria nuestra experiencia a la nueva situación y aprendiendo de ella para casos posteriores.

Los sistemas de razonamiento basados en casos funcionan de manera muy similar al razonamiento explicado arriba. Una definición para sistemas de este tipo es la dada por Riesbeck y Schank en 1989 y que recordamos a continuación,

> *Un razonador basado en casos resuelve nuevos problemas adaptando las soluciones que fueron utilizadas para resolver problemas previos similares*

Historia del CBR

El CBR aparece como alternativa a los sistemas basados en reglas. Sus raíces están en el trabajo de Roger Schank [31] y sus estudiantes en la Universidad de Yale en los Estados Unidos que sentó las bases de los sistemas basados en casos a principios de los años 80. Tras esto aparecen más sistemas que usan el CBR como es el sistema Protos desarrollado por Bruce Porter y su grupo de la Universidad de Texas, aplicado al diagnóstico de enfermedades del aparato auditivo, o el trabajo de Edwina Rissland y su grupo de la Universidad de Massachussets, Hypo, que se aplica al dominio del sistema judicial americano. Este sistema daba argumentos a favor y en contra basándose en casos legales anteriores. La primera aplicación en la industria fue en el sistema CLAVIER. Actualmente el CBR está siendo utilizado en aplicaciones help-desk, es decir, en servicios de soporte técnico.

Adquisición de Casos

Como se puede deducir de la definición de razonamiento basado en casos la parte fundamental de estos sistemas, y en muchas ocasiones el principal cuello de botella, son los casos sobre los cuales se va a razonar. Aun así, es mucho más sencillo adquirir el conocimiento necesario para el razonamiento en sistemas CBR que para otro tipo de sistemas como por ejemplo, los basados en reglas.

Una definición para *caso* en los sistemas CBR es la dada por Kolodner y Leake en 1997,

> *"Un caso es un fragmento contextualizado de conocimiento que representa una experiencia y que enseña una lección importante para conseguir los objetivos del razonador"* [34]

Para que un sistema CBR comience a funcionar es suficiente con tener varios problemas resueltos, es decir, varios casos sobre el dominio de trabajo. Para conseguirlos muchas veces no será necesario estar en contacto con un experto, cosa que si que ocurre si lo que quisiésemos es un sistema basado en reglas puesto que en este caso es necesario formalizar todo el conocimiento en reglas que debe proporcionar un experto. Esto no es nada sencillo ya que el ingeniero del conocimiento debería estar en contacto continuo con el experto y cabe la posibilidad, según la dificultad del dominio, que el ingeniero no sea capaz de crear reglas que se ajusten a la realidad del dominio. Con los casos esto se simplifica puesto que estos se pueden conseguir, por ejemplo, de registros que se realicen sobre el dominio. De esta manera se construye la base de casos sobre la que se razonará y se completará gracias a resultados de razonamientos posteriores. La base del conocimiento construida podrá tener tanto casos positivos como negativos, siendo estos últimos los que avisen sobre posibles fallos o excepciones.

El ciclo CBR

El razonamiento basado en casos sigue un ciclo de cuatro etapas conocido por las 4 R's (del inglés: Retrieve-Reuse-Revise-Retain). A continuación vamos a explicar una a una en que consisten dichas etapas.

- **Recuperación** (*Retrieve*)

  Tras recibir un problema que queremos solucionar, el primer paso de un sistema con razonamiento basado en casos es buscar en la base de conocimiento un caso que se parezca al problema actual. Para ello debe de haber una función de similitud que permita comparar el caso actual con los casos almacenados en la base de conocimiento y que determine cuál es el que más se ajusta a lo que queremos solucionar.
  Existen para la recuperación dos posibilidades. La primera de ellas consiste en almacenar los casos en alguna estructura de datos, como un grafo, de manera que los casos se recuperen siguiendo una cierta función de similitud. La segunda posibilidad consiste en una búsqueda lineal, comparando con todos los casos cuál es el que más conviene para el problema actual. Aunque este método puede ser muy ineficaz sobre todo en bases con muchos casos. En general, el proceso de recuperación va a depender mucho de la manera en la que esté estructurada la base del conocimiento.

Por lo tanto, hay que construir una función de similitud que permita seleccionar lo importante de la consulta, se decir, del problema actual, para compararlo con los casos almacenados y seleccionar los posibles candidatos a ser solución del problema actual. Puede darse el caso, de hecho es muy probable, sobre todo si hay muchos casos en la base de conocimiento de que haya más de un caso que se ajuste a la consulta de entrada. En ese caso, hay dos posibilidades: la primera sería seguir el criterio de la función de similitud anterior y escoger el caso con mayor función de similitud o hacer un segundo análisis en el cual se usarían nuevos criterios para escoger de entre los recuperados el mejor candidato de todos.

- **Reutilización** *(Reuse)*

  El caso recuperado de la base de conocimiento se reutiliza para resolver el problema actual. Al reutilizar un caso puede ocurrir que la solución dada por el sistema pueda ser utilizado tal cual sin hacer ningún tipo de cambio, es decir, el caso tal cual sirva como solución para el problema actual. Sin embargo, en muchas ocasiones aunque el caso recuperado sea el de mayor similitud es posible que necesite ciertas adaptaciones para poder convertirse en solución del problema. En ese caso se puede dar el caso tal cual al usuario y que esté sea él que lo adapte o lo interprete según su consulta. También puede ser el propio sistema que identifique cuáles son las cosas de la consulta que no cuadran con el caso recuperado y se encargue de adaptar la solución, simplemente sustituyendo unas cosas por otras o haciendo cambios más complejos.

- **Revisión** *(Revise)*

  Esta es la tercera etapa del ciclo CBR y no se implementa en todos los sistemas CBR. Sólo se lleva a cabo en algunos sistemas basados en casos cuando la solución propuesta no es correcta. Para poder saber si la solución propuesta es correcta o no se necesita evaluarla de alguna manera, bien por la evaluación de la solución por parte de un experto o probando la solución dada en una simulación. Si tras esto se observa que la solución es incorrecta se debería pasar a una fase de reparación para arreglar las cosas que no estuviesen bien y que la solución fuese correcta.

- **Aprendizaje** *(Retain)*

  Esta es la última fase del ciclo y muy importante en los sistemas basados en casos. Al terminar las fases anteriores el resultado obtenido se almacena en la base de conocimiento de manera que el sistema tiene un nuevo caso que puede ser usado en cualquier otra consulta. De esta manera, el sistema y los resultados de las consultas mejoran con el tiempo ya que la base de casos del sistema se va ampliando cada vez que se usa el sistema. También la eficiencia del sistema mejora gracias a esta fase.

1.3.2 Adaptación Constructiva

El método de razonamiento explicado más arriba es un método transformacional ya que cada vez que se busca una solución a una consulta o problema utilizamos un caso existente en la base de casos y posteriormente se transforma, si es necesario, para solucionar mejor el problema de entrada. Sin embargo, existen otros métodos que en lugar de transformar una solución existente lo que hacen es ir generando una solución nueva a partir de casos almacenados, estos sistemas construyen la solución poco a poco. Uno de estos métodos, que explicamos a continuación, es el de *Adaptación Constructiva* [18]. Esta técnica, en términos abstractos, puede entenderse como una búsqueda en el espacio de soluciones donde los casos son usados en dos fases principales: la fase de generación de hipótesis y la fase de ordenación de hipótesis.

La adaptación constructiva (CA) es una forma de búsqueda heurística primero el mejor en el espacio de soluciones que usa información de los casos para guiar la búsqueda. Se han desarrollado diferentes modalidades de adaptación constructiva: en alguna de ellas el proceso de búsqueda es exhaustivo, es decir, se busca entre todos los casos existentes, en otros no es exhaustivo, en algunos la representación de casos y de estados es idéntica, en otros no lo es.

Para comenzar definimos un caso *Ci = (Pi, Ki)* como una tupla de descripción del problema *Pi* y una solución *Ki*. En la descripción del problema *Pi* están incluidos los requerimientos del problema *Req (Pi).* Normalmente los requerimientos del problema se corresponden con lo que ha sido introducido por el usuario. Puesto que la solución es una configuración, podemos considerar que la configuración es una estructura de esos conceptos y relaciones. Denotamos *k* el conjunto de posibles configuraciones incluyendo tanto configuraciones parciales como completas. Además vamos a denotar el conjunto de configuraciones completas como *kc* que está incluido en el conjunto *k*. Una solución para un caso debe de ser tanto completa como válida. Una configuración *ki* es completa cuando *ki* pertenece a *kc*. Además, decimos que una solución es válida cuando *sat* (Req (Pi, *ki*)), es decir, si la solución satisface los requerimientos de entrada. La adaptación constructiva trabajo sobre estados. Un estado es una representación de dominio especifico de la información necesitada para representar parcialmente la solución, en nuestro caso una configuración parcial. Un estado va a contener más información que la configuración parcial, tiene que contener los requerimientos de entrada del usuario, los valores intermedios usados para la resolución del problema…siendo conveniente implementar la representación del estado distinta de la representación del caso.

El proceso de adaptación constructiva se basa en una búsqueda primero el mejor con dos funciones básicas: la generación de hipótesis y la ordenación de hipótesis, y dos funciones auxiliares: función objetivo y estado inicial. Para ello se puede usar cualquier algoritmo de búsqueda primero el mejor como por ejemplo el algoritmo A*.

La función estado inicial es una función que crea un estado inicial, por el que se comenzará la búsqueda y la adaptación constructiva. La función objetivo es una función simple cuya función es simplemente determinar si dado el estado actual, este es un estado objetivo y se puede detener el algoritmo porque ese estado es solución al problema o si se debe continuar con la búsqueda ya que todavía no se ha llegado a una solución para el problema.

Generación de Hipótesis

Esta es una de las principales funciones que debe de tener el algoritmo usado en la adaptación constructiva. Para ello se necesita el conocimiento adquirido en los casos. En esta etapa de la construcción de la solución, dado un estado abierto, este se expande generando el conjunto de sus sucesores *s*. La idea esencial de la generación de hipótesis es que en el caso de que existiesen varias opciones acerca de elementos o relaciones que sean satisfactibles con el conjunto de estados *s* entonces estas opciones son consideradas como posibles hipótesis y se crea un nuevo estado sucesor que es añadido al conjunto hipótesis. Después de esta etapa es necesario escoger entre todos los nodos sucesores cuál será el siguiente a expandir para terminar alcanzando la solución final. Para ello se pasa a una nueva fase, la fase de ordenación de hipótesis.

Ordenación de Hipótesis

Una vez que se han expandido todos los sucesores debemos ordenarlos para poder quedarnos con el mejor de todos los nodos abiertos. De esto se encarga la función de Ordenación de hipótesis. Esta función, al igual que la anterior, depende del dominio en el que estemos trabajando y por tanto las cosas a tener en cuenta en cada una de las fases deberán ser definidas dependiendo del sistema en el que nos encontremos. Una vez ordenadas las hipótesis, el algoritmo selecciona la mejor de ellas para continuar la construcción del sistema o terminando la construcción en caso de que el estado seleccionado se corresponda con un estado objetivo.

Cuando se termina la construcción de la solución se hace como en el ciclo CBR transformacional pudiendo el sistema aprender del nuevo caso construido almacenándolo en la base de conocimiento para poder ser reutilizado más adelante.

Un ejemplo de Adaptación Constructiva

Para que quede más claro el funcionamiento de la adaptación constructiva vamos a dar un ejemplo de sistema que usa este tipo de adaptación. Se trata del *SaxEx*, un sistema cuyo razonamiento se corresponde con la adaptación constructiva y que tiene como fin el de generar melodías basadas en ejemplo de interpretaciones humanas que conforman la base de casos.

La entrada de este sistema es una frase musical en formato MIDI[6] acompañada por algunas etiquetas que permiten definir el ritmo de la interpretación. Una solución para un problema en *SaxEx* es una estructura secuencial de notas pertenecientes a una frase musical, es decir, una sección breve de una composición que tiene sentido propio por ella misma. Cada nota de la frase tiene asociado un *modelo expresivo* compuesto por varios parámetros (amplitud del sonido, duración de las notas, frecuencia del vibrato y amplitud del vibrato de las notas...entre otros). En definitiva, se alcanza una solución cuando se consigue un valor para cada parámetro.

[6] MIDI son las siglas de Musical Instrument Digital Interface (Interfaz Digital de Instrumentos Musicales). Se trata de un protocolo industrial estándar que permite a las computadoras, sintetizadores, secuenciadores, controladores y otros dispositivos musicales electrónicos comunicarse y compartir información para la generación de sonidos.

Un estado en *SaxEx* contiene información acerca de: los requerimientos de entrada, los modelos expresivos generados hasta el momento y el conjunto de *notas abiertas*, es decir, notas que están en la frase pero que todavía no tienen un modelo expresivo completo asignado. El estado inicial esta formado por la entrada al sistema y todas las notas de la frase música.

En la fase de generación de hipótesis, dado un estado *s* selecciona la siguiente nota entre las notas abiertas y usa casos para generar varios modelos expresivos para la nota, cada una expresada como un nuevo estado sucesor.

La fase de ordenación de hipótesis, usa conocimiento de dominios y de casos. El conocimiento de dominio valora la coherencia de los diferentes modelos expresivos en un estado. El conocimiento de casos es usado para estimar un valor de similitud global de un estado respecto a los casos usados para generar este estado. Esto se da añadiendo unos valores de similitud a las notas con modelos expresivos pertenecientes al estado. La ordenación de hipótesis combina estos dos valores en un valor de estado y ordena los estados abiertos según estos valores.

Por último, la función objetivo sólo se encarga de verificar que la solución sea completa y valida. La validación en *SaxEx* está definida por dos valores umbrales de suavidad y variación que deben ser satisfechos en el estado final. Estos valores pueden ser elegidos por el usuario según su interés musical personal.

1.3.2.1 Algoritmo A*

La adaptación constructiva se basa en la aplicación de un algoritmo de búsqueda heurística. El algoritmo A* (*A estrella*) se clasifica dentro de los algoritmos de búsqueda en grafos. Presentado por primera vez en 1968 por Peter E. Hart, Nils J. Nilsson y Bertram Raphael, el algoritmo encuentra, siempre y cuando se cumplan unas determinadas condiciones, el camino de menor coste entre un nodo origen y uno objetivo dentro del espacio de búsqueda compuesto por todos los posibles nodos que se pueden desarrollar para un problema concreto y las conexiones entre ellos.

Características del algoritmo A*

El algoritmo A* utiliza una función de evaluación $f(n) = g(n) + h(n)$, donde $h(n)$ representa el valor heurístico del nodo a evaluar, y $g(n)$, el coste real del camino recorrido para llegar a dicho nodo. A* mantiene dos estructuras de datos auxiliares, que podemos denominar *abiertos*, implementado como una cola de prioridad (ordenada por el valor $f(n)$ de cada nodo), y *cerrados*, donde se guarda la información de los nodos que ya han sido visitados. En cada paso del algoritmo, se expande el nodo que esté primero en abiertos, y en caso de que no sea un nodo objetivo, calcula la $f(n)$ de todos sus hijos, los inserta en abiertos, y pasa el nodo evaluado a cerrados.

Función de evaluación o función heurística

El algoritmo A* se basa en el resultado de la función de evaluación o heurística que se aplica a cada uno de los nodos del grafo resultante del espacio de búsqueda del problema en cuestión. Es posible construir una función de evaluación de estados que incluya el coste para alcanzar un estado determinado así como una aproximación sobre el coste para encontrar el objetivo desde ese estado como se muestra en la figura 1. Se definen los siguientes costes:

- **f(n)'** es el coste heurístico asociado a un estado n
- **g(n)** es el coste real para alcanzar un estado n a partir del estado inicial
- **h(n)'** es el coste heurístico para alcanzar un objetivo desde el estado n

La relación que existe entre los costes es:

f(n)' = g(n) + h(n)'

Una función heurística h es admisible si las estimaciones nunca superan los valores reales de los estados evaluados, es decir, f(n)'≤ f(n). La admisibilidad es una propiedad importante de la función heurística, y frecuentemente es requerida por los algoritmos de control para garantizar un funcionamiento correcto.

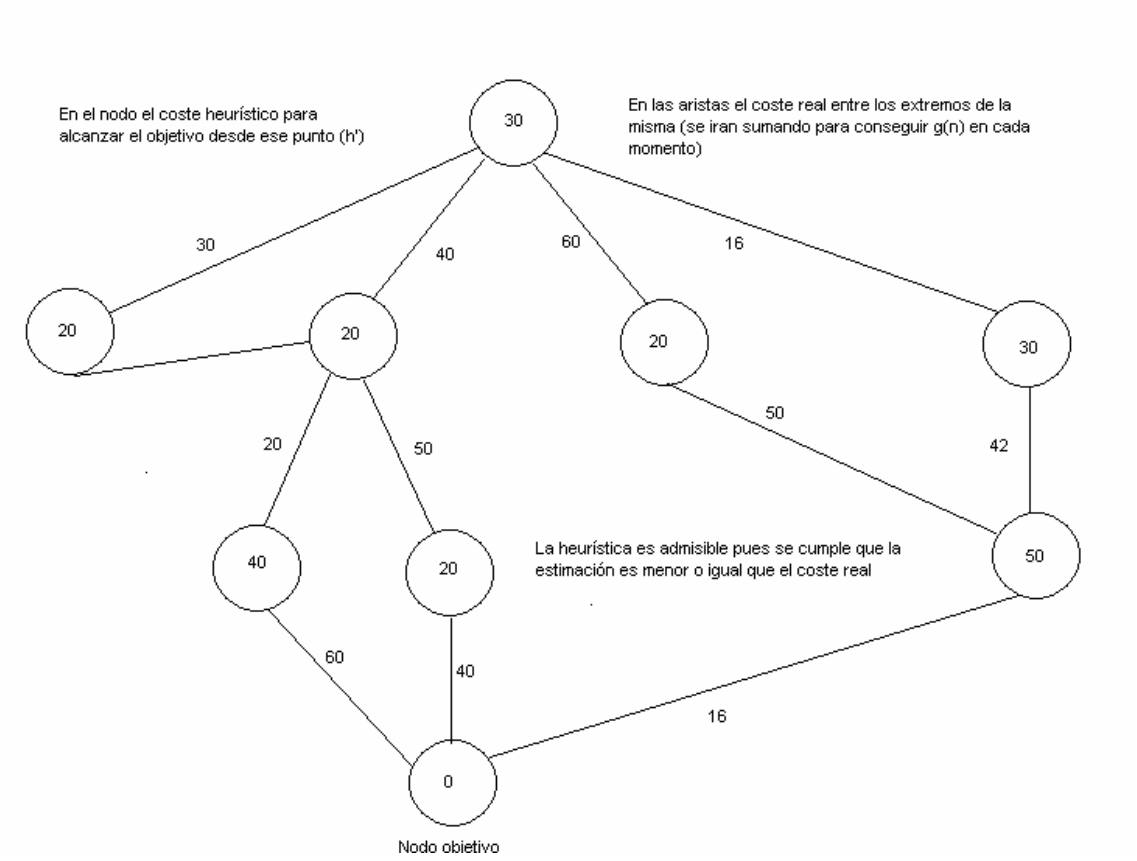


Figura 1.2: Ejemplo de grafo para aplicar A* con heurística admisible

Propiedades del algoritmo A*

El algoritmo A* tiene las siguientes propiedades:

1. Como todo algoritmo de búsqueda en anchura, A* es un algoritmo completo, lo que quiere decir que en caso de existir una solución, siempre dará con ella.
2. Si para todo nodo n del grafo se cumple $g(n) = 0$, nos encontramos ante una búsqueda voraz. Si para todo nodo n del grafo se cumple $h(n) = 0$, A* pasa a ser una búsqueda de coste uniforme no informada.
3. Para garantizar la optimalidad del algoritmo, la función $h(n)$ debe ser admisible, esto es, que no sobrestime el coste real de alcanzar el nodo objetivo.
4. De no cumplirse la condición anterior, el algoritmo pasa a denominarse simplemente A, y a pesar de seguir siendo completo, no se asegura que el resultado obtenido sea el camino de coste mínimo. Asimismo, si garantizamos que $h(n)$ es consistente (o monótona), es decir, que para cualquier nodo n y cualquiera de sus sucesores, el coste estimado de alcanzar el objetivo desde n no es mayor que el de alcanzar el sucesor más el coste de alcanzar el objetivo desde el sucesor.

Complejidad computacional

La complejidad computacional del algoritmo está íntimamente relacionada con la calidad de la heurística que se utilice en el problema. En el caso peor, con una heurística de pésima calidad, la complejidad será exponencial, mientras que en el caso mejor, con una buena $h(n)$, el algoritmo se ejecutará en tiempo lineal. Para que esto último suceda, se debe cumplir que $h(x) \leq g(y)-g(x) + h(y)$. Donde h* es una heurística óptima para el problema, como por ejemplo, el coste real de alcanzar el objetivo.

Complejidad en memoria

El espacio requerido por A* para ser ejecutado es su mayor problema. Dado que tiene que almacenar todos los posibles siguientes nodos de cada estado, la cantidad de memoria que requerirá será exponencial con respecto al tamaño del problema.

1.4 Arquitectura del sistema

Para conseguir los tres objetivos principales propuestos podemos distinguir tres partes bien diferenciadas en la arquitectura de nuestro sistema. Dichas partes son:

- Editor de cuentos, que se encarga de transformar unos datos de entrada en una representación válida para el conocimiento de nuestro sistema. Así conseguimos la adquisición de conocimiento para nuestro sistema. Esta parte se explica detalladamente en el capítulo 3.

- Razonador, que a partir de los datos contenidos en la base de conocimiento genera la estructura del guión de un cuento. Todo lo relacionado con el razonamiento de nuestro sistema se detalla en el capítulo 4.

- Módulo del lenguaje natural, que se corresponde con el capítulo 5 de esta memoria, que a partir del guión de un cuento genera una representación del mismo en lenguaje natural, es decir en frases legibles para el usuario.

La figura 1.3 muestra los diferentes módulos del sistema así como su interconexión y resultados.

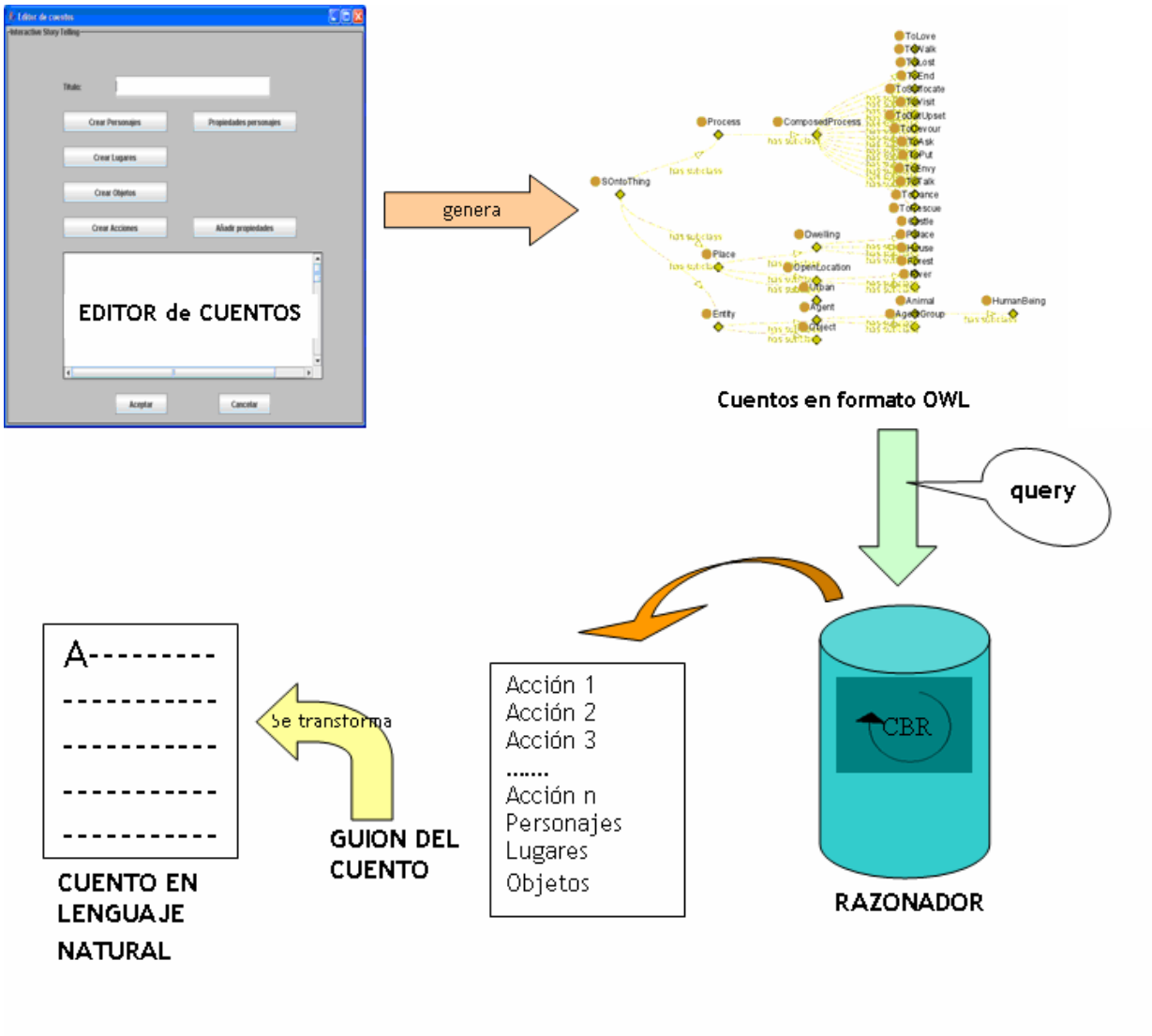


Figura 1.3: Visión general del sistema.

1.5 Herramientas utilizadas

Para la implementación del sistema hemos necesitado diferentes programas y librerías. Todos ellos son código abierto y han sido descargados de Internet.

- **Java** [20]

  El lenguaje elegido para implementar el sistema por su portabilidad, legibilidad y excelentes propiedades como lenguaje orientado a objetos.

- **Protégé 3.2** [19]

  Se trata de una herramienta para el desarrollo de ontologías y sistemas basados en el conocimiento que ha sido creado en la Universidad de Stanford. Está desarrollada en Java.

  Protégé emplea una sencilla interfaz de usuario que permite crear fácilmente las clases, instancias y propiedades que componen una ontología. Además, Protégé ofrece diferentes plug-ins para completar su funcionalidad. En particular, hemos usado Jambalaya, desarrollado por el grupo de ingeniería del software CHISEL de la Universidad de Victoria, que permite crear gráficos para ver la ontología creada por el usuario.

- **OntoBridge 1.2** [21]

OntoBridge[7] es una librería, desarrollada por el grupo de investigación GAIA (Grupo de Aplicaciones de Inteligencia Artificial) de la Facultad de Informática de la Universidad Complutense de Madrid, con el fin de manejar ontologías de una forma fácil. Está basado en Jena 2.4 y es forma parte de jCOLIBRI[8].

OntoBridge permite trabajar con ontologías localmente u online, permite el uso de Pellet como razonador, incluye métodos para acceder y modificar conceptos, instancias o propiedades en la ontología así como para guardar los cambios realizados en ficheros OWL RDF/XML.

En nuestro sistema, OntoBridge ha permitido, la comunicación entre el programa Java y la ontología. De esta manera se ha podido realizar las recuperaciones necesarias para llevar a cabo el razonamiento, así como para poder introducir nuevos cuentos en la ontología desde una interfaz sencilla.

- **Eclipse 3.2** [22]

Eclipse es una plataforma de software de código libre. Fue desarrollado inicialmente por IBM pero en la actualidad está siendo desarrollado por la Fundación Eclipse, organización independiente sin ánimo de lucro que fomenta una comunidad de código libre.

Hemos usado Eclipse, con el JDK 5.0, como editor y compilador del código Java de nuestro sistema.

[7] OntoBridge está bajo licencia LGPL y ha sido desarrollado por Juan A. Recio-García.
[8] http://gaia.fdi.ucm.es/grupo/projects/jcolibri/index.html

- **JSL: Java Search Library** [23]

En nuestro sistema necesitábamos para la generación de cuentos un algoritmo de búsqueda heurística. Por esta razón incluimos la librería JSL, desarrollada en Java, que proporciona algoritmos generales de búsqueda en grafos. El dominio de búsqueda debe ser implementado por el usuario.

2. TRABAJO RELACIONADO

Aunque existen muchas teorías sobre narración interactiva, es decir sobre la generación de historias, sólo comentaremos algunas de las que han servido de inspiración para nuestro proyecto.

2.1 Morfología de un cuento

La base de nuestro trabajo son las ideas de Vladimir Propp [1] sobre los cuentos populares rusos. Propp analizó estos cuentos hasta que encontró una serie de puntos recurrentes que creaban una estructura constante en todas estas narraciones. Es lo que se conoce como *las funciones de Propp.* Esto dota a los cuentos populares de una descripción acorde a las partes que los constituyen, las relaciones entre esas partes y las relaciones de esas partes con el total. La idea principal es que los cuentos populares son creados a partir de ingredientes que cambian de un cuento a otro y de ingredientes que no cambian. Según Propp, entre distintos cuentos lo que cambian son los nombres -y ciertos atributos- de los personajes, mientras que sus acciones permanecen iguales, pero no en todos los cuentos hay todas las acciones. Es decir, no son iguales. Estas acciones que actúan como constantes en la morfología de los cuentos populares es lo que él define como funciones. Algunos ejemplos de las funciones de Propp son: *Villainy (acto de villanía), Departure (partida por parte de algún personaje), Interdiction (prohibición), Interdiction Violated (prohibición violada), Acquisition of a Magical Agent (adquisición de un objeto mágico), Testing of the hero (prueba del héroe)*, etc… Existen algunas restricciones a la hora de elegir las acciones que compondrán una nueva historia, dichas restricciones vienen dadas por las dependencias implícitas entre funciones: por ejemplo, para poder aplicar la función, Interdiction Violated, el héroe debe haber recibido una orden (función Interdiction).

Proyectos que emplean la morfología de Propp

Muchos son los proyectos que utilizan la morfología de Propp a la hora de construir sus historias.

OPIATE [11]: un generador interactivo que crea nuevas historias reutilizando los cuentos, analizados en términos de funciones de Propp, similares a las ideas del diseño de una historia más convencional.
PftML [12] (Proppian fairy tale Markup Language) es un proyecto que pone un DTD (Document Type Defnition) en ejecución para estandardizar un modelo analítico formal para los cuentos basados en Propp.
Otro ejemplo es, The Proppian Fairy Tale Generador [13], un generador al azar simple, que utiliza las funciones de Propp para generar nuevos cuentos, encadenando partes de texto escrito. Otros sistemas clásicos son Automatic Novel Writer [15], Joseph [16] y por supuesto Minstrel [8] inspirados en el trabajo original de Propp.

2.2 Narración interactiva

Otro trabajo muy importante, y que fue el principio de nuestro proyecto es KIIDS [14] (Knowledge-Intensive Interactive Digital Storytelling system).
KIIDS es un sistema realizado con el fin de integrar dirección automática en el desarrollo de aplicaciones de narración digital e interactiva de forma genérica y reutilizable. El sistema consiste en un repositorio extensible de ontologías y un armazón software de componentes reutilizables. Las ontologías representan parcelas de conocimiento sobre interacción, narración y simulación, y los componentes implementan técnicas específicas que servirán para generar los elementos de las historias específicos para cada dominio y aplicación. ProtoPropp[9] es una aplicación de KIIDS que nos ofrece generación automática de cuentos populares rusos.

ProtoPropp es una aplicación de Knowledge-Intensive Case-Based Reasoning (KI-CBR) para resolver el problema de la generación de historias a partir de una base de casos de cuentos analizado en términos de las funciones de Propp. El proceso CBR para la generación de historias está definido a partir de una query[10] específica de usuario inicial, con distintas configuraciones para las historias, empleando una ontología que nos permite medir la distancia semántica entre palabras y las estructuras que forman parte del texto. Esto constituye un avance en trabajos previos, similares soluciones habían sido aplicadas en la generación de poesías empleando técnicas CBR pero utilizando sólo información sintáctica durante el proceso de adaptación. ProppOnto es la ontología de ProtoPropp implementada en OWL con cada función narrativa definida por Vladimir Propp. Estos trabajos fueron el comienzo de lo que sería nuestro proyecto, de ProtoPropp surgieron las ideas de basarnos en la morfología de los cuentos de Propp, y de ProppOnto, la ontología de ProtoProp, nos sirvió de base a lo que sería nuestra ontología, ya que en esta ya estaban organizadas las distintas funciones de Propp así como la jerarquía de personajes y lugares, aunque después nosotras refináramos acorde a nuestras necesidades dicha jerarquía.

También forma parte de la investigación realizada en nuestro proyecto, el trabajo de Andrew Gordon, dedicado, en la actualidad, a la investigación interdisciplinaria en la Universidad de California Meridional en el Instituto de Tecnologías Creativas (ICT) en las áreas de la inteligencia artificial, del proceso de lenguaje natural y de la interacción humano-computadora. Sus investigaciones se centran en diferentes ámbitos de entre los que se pueden destacar los siguientes: Historias basadas en ambientes de aprendizaje, Comprensión de idiomas naturales. De todo su trabajo, aquél que ha servido para el nuestro han sido las ideas que se explican en el artículo *Experience Management Using Storyline Adaptation Strategies [3]*. El artículo discute sobre la creación de un sistema que reaccione a una serie de decisiones tomadas por el usuario de manera que siempre reconduzca la situación hacia los objetivos que busca dicho sistema. La idea está aplicada a entornos militares. Parte de la idea de la creación de historias interactivas partidas en momentos. Al final de cada uno de los momentos el usuario se encuentra con una toma de decisión, de manera que podríamos pensar que en este punto la historia, continuaría de diferente manera de acuerdo a lo que el usuario elija. Sin

[9] http://www.fdi.ucm.es/profesor/fpeinado/projects/kiids/apps/protopropp/
[10] Entendemos por query una estructura de datos que almacena la información necesaria para producir un paso del razonamiento

embargo, el sistema es capaz de ir reconduciendo la historia en caso de que el usuario no elija la opción esperada.

Que el sistema sea capaz de reconducir la historia no es algo fácil y sencillo. Podríamos representar nuestro problema como una matriz NxM, donde N es el número de momentos de la historia, y M el número de opciones en cada momento. De esta manera tendríamos, que dicho sistema debería tener NxM adaptaciones de la historia para que nunca se desviara de los objetivos iniciales. Esto es un problema bastante importante cuando la historia y las opciones a lo largo de la misma son elevadas y hace que la aproximación sea impensable.

Durante el resto del artículo hay una explicación a la solución adoptada para resolver el problema.

Las ideas obtenidas de este trabajo son bastante interesantes, puesto que trata del tratamiento de historias interactivas con los usuarios de la aplicación. En nuestro caso decidimos no optar por reconducir la historia si la opción elegida por el usuario no era la esperada, sino que la historia continuara en función de la elección, ya que en principio no existe ninguna expectativa inicial de cómo debe terminar el cuento que genere nuestro sistema. En cuanto a las representaciones sugeridas por Gordon, decidimos contar con el formalismo basado en ontologías, explicado más adelante, ya que presenta un gran potencial para la representación de historias.

Siguiendo la idea de representación ontológica podemos destacar el sistema *Minstrel* de *Scout R.Turner*, consistente en una aplicación de generación automática de pequeñas historias ambientadas en el mundo del rey Arturo y la mesa redonda. Turner, se basó en heurísticas creativas y métodos de resolución de problemas para hacer ver cómo, mediante simples, pero poderosos mecanismos de razonamiento y manipulación de gran cantidad de conocimiento, se pueden obtener sistemas inteligentes.

La representación de conocimiento se basa en *frames*, esquemas con *slots* y *facetas*, para representar los estados del mundo, las acciones de los personajes, los planes, los beneficios, los efectos, etc... Las relaciones se representan con operadores semánticos y algoritmos definidos empíricamente.

Desde el punto de vista de la ingeniería del conocimiento el sistema presentaba los siguientes problemas:

1. Era un sistema difícil de mantener.
2. El sistema dificultaba la reutilización.
3. El sistema era de difícil adaptación a otros proyectos.

Estos y otros motivos son los causantes de plantear la adaptación del sistema a una nueva forma de trabajo, la propuesta por el W3C [11](OWL).

[11] El propósito de W3C de formalizar ontologías para la interoperatividad en la Web, dio lugar a OWL. Podemos destacar la rama OWL LD designado a aplicaciones de mucha expresividad sin abandonar la base computacional. Se basa en descripciones lógicas (lógica de primer orden).

La implementación del sistema *Minstrel* adaptada a OWL tiene las siguientes características:

1) Representa los esquemas de conocimiento del sistema original en una moderna ontología y base de conocimiento
2) Sólo se basa en el universo del rey Arturo y de la mesa redonda.
3) Conceptos e individuos se separan en dos campos: Minstrelonto y Minstrelkb.
4) La prioridad de metas deja de ser trivialmente restringida mediante la enumeración (1...100) como en el sistema original.
5) Los archivos con la ontología y la base de conocimiento están realizados en Protégé-owl usando Pellet como razonador para validar el modelo.

Por lo tanto, siguiendo el modelo de Turner, decidimos utilizar Protégé-OWL usando Pellet como razonador para nuestra ontología. También nos planteó el reto de crear una ontología con una estructura sólida.

Otro trabajo que es interesante mencionar corresponde con el realizado por *Marc Cavazza, Fred Charles, y Steven J. Mead*, de la universidad de Teesside, Reino Unido. En concreto hablaremos sobre el artículo: *Character-Based Interactive Storytelling [10]*.

Su aproximación a la hora de trabajar con la creación de historias interactivas se basa en la definición de una línea argumental para cada personaje, el cual además está definido con una serie de características que el usuario, en cualquier momento, puede ampliar o modificar mediante interacciones con el sistema. Para implementar la representación de los personajes han utilizado HTN (Hierarchical Task Networks). Podemos pensar que HTNs es una representación implícita del conjunto de posibles soluciones a un determinado objetivo. Es decir, un personaje se mueve por objetivos. Tiene un objetivo principal que se puede descomponer en sub-objetivos, y esto es lo que se representa mediante HTN (gráficamente es un árbol). Durante el artículo se explica cómo se ha realizado la planificación de la historia a través de las HTNs y cómo éstas van cambiando en función de las acciones del usuario.

El artículo concluye definiendo cómo entiende este sistema una historia: *Una historia consiste en la conjunción de un conjunto de acciones terminales tomadas de cada uno de los planes de los personajes.*

Este artículo nos planteó la posibilidad de recoger datos del usuario a la hora de generar la historia dándole a elegir un personaje, y que dicho personaje tenga asociado un objetivo relacionado con el final de la historia. Pero como veremos en el desarrollo del proyecto, esta idea se deja como trabajo futuro.

Por último comentaremos el trabajo de R. Michael Young, profesor asociado del departamento de ciencia computacional de la Universidad de Carolina del Norte. Sus intereses giran alrededor del uso de la inteligencia artificial y sus técnicas, en el mundo virtual de los videojuegos. En la actualidad sus investigaciones de centran en los ámbitos de generación de lenguaje natural, videojuegos, y modelos computacionales narrativos.

A continuación analizamos las ideas de uno de sus trabajos, *Notes on the Use of Plan Structures in the Creation of Interactive Plot [2]*, muy relacionado con el tema de la generación de historias.

Para Young la interfaz de un espacio virtual presenta una representación artificial del mundo y de las actividades dentro de él. Cuando un usuario entra en un espacio virtual se debe crear un "contrato" entre el usuario y el autor del sistema. Este contrato debe establecer la expectativa de que el sistema obedezca a ciertas convenciones narrativas. El contrato debe ser beneficioso para los dos: diseñador y usuario, por un lado facilita al usuario la comprensión mientras da al diseñador unas restricciones en el vocabulario con el cual el diseñador construye los eventos que ocurren en el mundo virtual. Algunos diseñadores de sistemas eligen renunciar al control del usuario o pasar por alto gran parte de la mediación. Para los sistemas interactivos, la mediación es el punto de apoyo en el que se levanta el poder del modelo computacional narrativo.

Young propone un modelo de planificación diferente de otros en dos sentidos. Primero, la representación del guión que usa utiliza una rica representación formal de la estructura causal del guión. Cada dependencia causal entre el fin, la precondición y los efectos en el guión, están cuidadosamente definidos durante la construcción del mismo. Segundo, el proceso de construcción del guión es una búsqueda de principio a fin en el espacio de posibilidades.

En nuestro caso el guión de la historia se irá componiendo en base a una serie de propiedades de cada una de las acciones ya seleccionadas para la historia en creación, tal y como dice Young. Para la construcción de este guión, el sistema se basa en sucesivas búsquedas en el espacio de posibilidades que, en nuestro caso no se corresponde con toda la base de conocimiento, sino con un subconjunto que constituirá el espacio de posibilidades para cada uno de los diferentes estados del razonamiento del sistema.

Siguiendo con la interactividad Young plantea dos posibilidades, cada una con sus respectivos problemas. Una es eliminar toda intervención del usuario, cuyo resultado es un sistema reducido a una narrativa convencional como la de un libro o una película. La otra opción, es permitir que el usuario pueda tomar un control completo, en este caso, la coherencia narrativa de la interacción del usuario está limitada por su propio conocimiento y habilidad. Por esto muchos sistemas basados en narrativa toman un término intermedio entre los dos, especificando durante el diseño, momentos de la acción en los que el usuario puede intervenir en la narrativa. El conjunto de caminos narrativos resultantes está estructurado de manera que, cada camino da al usuario una interesante experiencia narrativa, lo cual limita el número de historias que se pueden contar.

Llegando a un equilibrio entre que el usuario tome por completo el control o que no puede intervenir en la generación de la historia, podemos conseguir un sistema aceptable y creativo. Esto fue lo que nos planteamos desde un principio, encontrar dicho equilibrio, de forma que el usuario pueda sentirse creador de su propia historia pero sin que llegara a tener un control total de la misma.

En la actualidad hay muchos sistemas intentando crear historias automáticas a través de metodologías muy distintas y que dificultan su comparación. El problema que existe en la actualidad es la creatividad computacional: crear cosas nuevas y diferentes al mismo tiempo. Para ello se presentan dos opciones. Por un lado tenemos sistemas con unas cuantas tramas principales que funcionan de modo que sólo modifican el contexto de la historia (no es buena aproximación desde el punto de vista de la narración), en el lado contrario está la generación de nuevas historias mediante la unión de eventos principales, que se desarrollan cronológicamente en la trama (es necesario una base de conocimiento previo).

Teniendo en cuenta que los objetivos de la creatividad son obtener un alto grado de originalidad y mantener coherencia narrativa, hay muchas teorías interesantes en el campo de la narración, pero algunas de ellas son difíciles de formalizar para traducirlas a una representación procesable en una máquina. Con toda esta información y todas estas ideas sobre narración interactiva comenzó nuestro trabajo. Este es el punto de partida desde el que ponemos en marcha nuestras teorías sobre cómo se pueden generar historias interactivas y creativas.

3. BASE DE CONOCIMIENTO DEL SISTEMA

Como hemos explicado anteriormente para nuestro proyecto funcionase correctamente había que decidir cómo representar todo el conocimiento del dominio de los cuentos infantiles en una ontología. Para el desarrollo de la nuestra tomamos como base la ontología KIIDSOnto, del sistema KIIDS. Una vez que se definió la manera de representar el conocimiento pudimos insertar algunos cuentos que formarían la base de casos del sistema. Finalmente, para poder facilitar la adquisición de casos implementamos una sencilla interfaz para que cualquier usuario pudiese insertar casos nuevos que mejorasen los resultados del sistema. En este capítulo se explica detalladamente todo esto.

3.1 Ontología

En este apartado vamos a explicar la estructura que tiene la ontología de nuestro sistema, las clases que la componen así como las propiedades definidas para poder especificar mejor los cuentos de la base de datos. Después de eso daremos un ejemplo concreto de cómo hacer una representación en nuestra ontología.

3.1.1 Estructura de la Ontología

En este apartado vamos a explicar la estructura de nuestra ontología. La ontología tiene una clase principal cuyo nombre es *ISTOnto* y que tiene dos subclases dividiendo la ontología en dos partes. La primera de ellas, de nombre *INSOnto*, es donde se encuentra el conocimiento del sistema. En esta parte están definidos todos los conceptos necesarios para especificar las historias y cuentos que forman la base de casos.

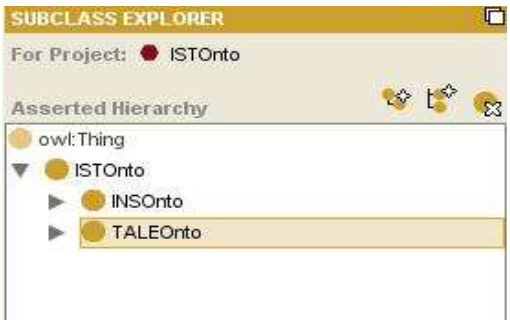


Figura 3.1: División de conceptos en ISTOnto

En todos los cuentos existen una serie de conceptos comunes, tales como el tipo de las acciones que ocurren en cada uno de ellos (funciones de Propp), los tipos de personajes (principal, secundario, etc.). En *INSOnto* podemos encontrar la definición de todos estas partes comunes, siguiendo la siguiente jerarquía.

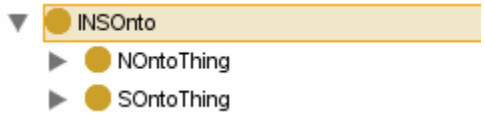


Figura 3.2: Jerarquía de INSOnto

- NOntoThing: representa los conceptos e identidades que podemos identificar de manera común en todos los cuentos, tales como la definición de las funciones de Propp, los tipos de personajes, los tipos de lugares, etc.
- SOntoThing: representa los conceptos e identidades específicas de cada cuento. Clasificación de los personajes (persona, animal, hombre, mujer…), de los lugares (bosque, ciudad, río…)

NOntoThing

Figura 3.3: Visión general de NOntoThing

Dentro de NOntoThing podemos separar los eventos (Event) y los conceptos (Existent). Los eventos se refieren a las funciones de Propp, que identifican las diferentes acciones de una historia. Mientras que, los conceptos existentes en una historia, identifican datos como, los tipos de rol[12] que representa un personaje en una historia.

[12] Entendemos por rol el papel que desempeña un personaje en un cuento (i.e: héroe, ayudante…). Estos roles también están definidos en la estructura de Propp.

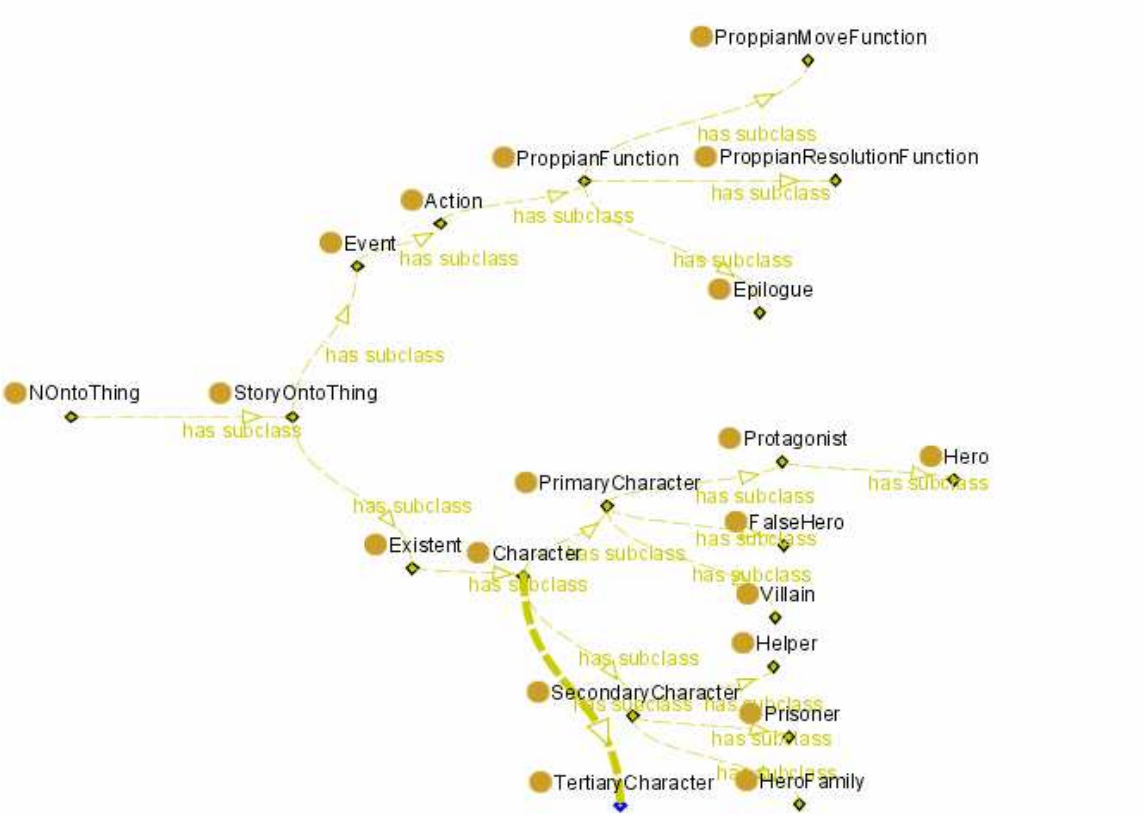


Figura 3.4: Gráfica que muestra las clases que componen NOntoThing

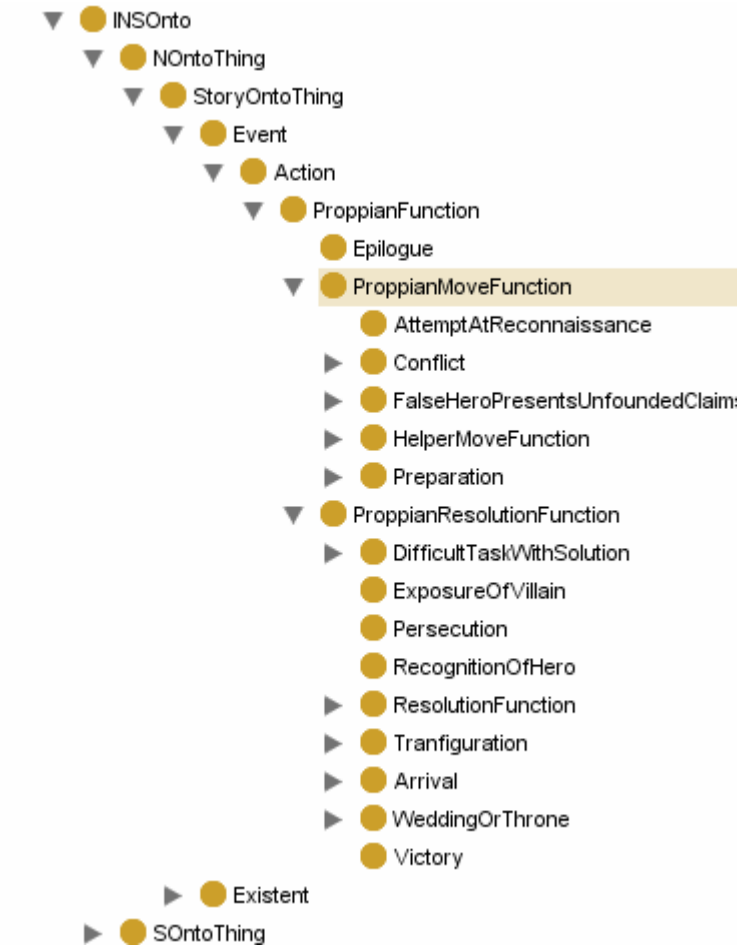


Figura 3.5: Visión general de las funciones de Propp.

Podemos observar la división de las funciones de Propp según la siguiente identificación:

- Epilogue: parte final a la trama de una historia.
- ProppianMoveFunction: trama de la historia.
- ProppianResolutionFunction: resolución y final de una historia.

Tanto en la figura 3.5 como en la figura 3.6 podemos ver una representación gráfica de estas funciones.

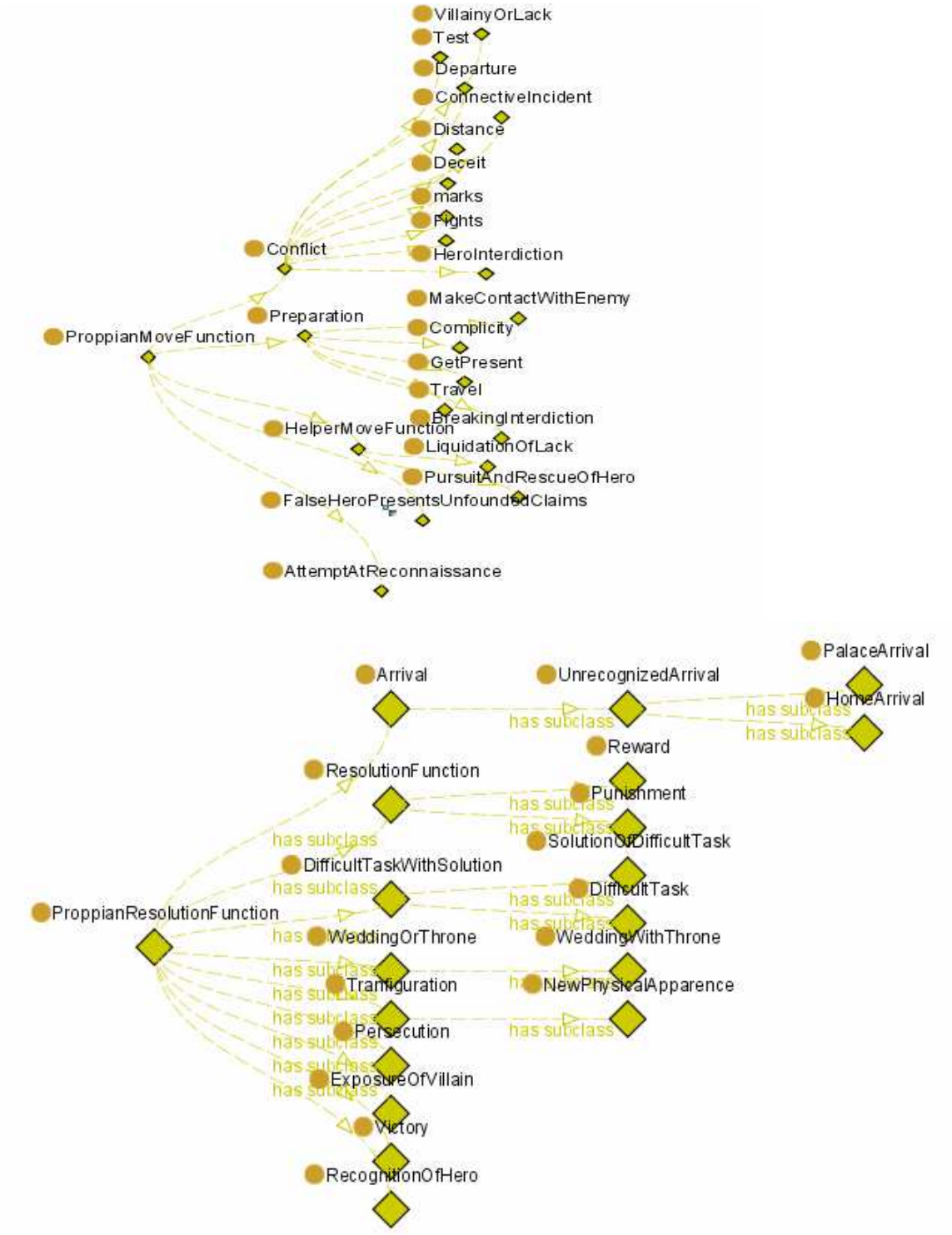


Figura 3.6: Representación gráfica de las funciones de Propp.

SOntoThing

Dentro de SOntoThing especificamos aquellos conceptos que son concretos de cada cuento. Identificaremos los personajes, los objetos, los lugares y algunas acciones no se están identificadas dentro de las funciones de Propp (i.e.: la acción de bailar, caminar…)

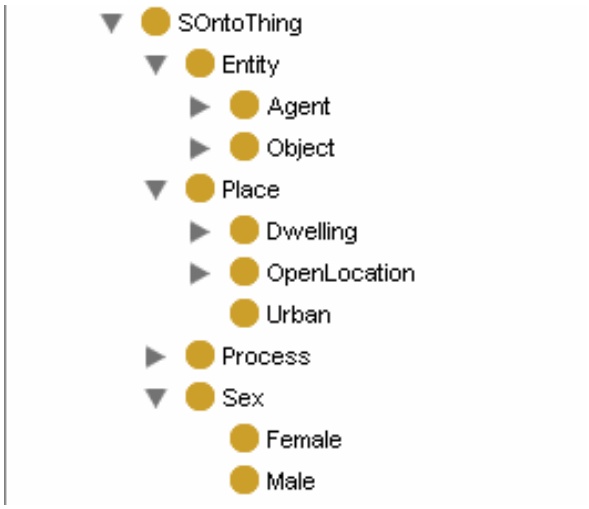


Figura 3.7: Conceptos que representan de manera concreta individuos de un cuento.

A continuación mostramos una gráfica obtenida que muestra las clases que componen SOntoThing

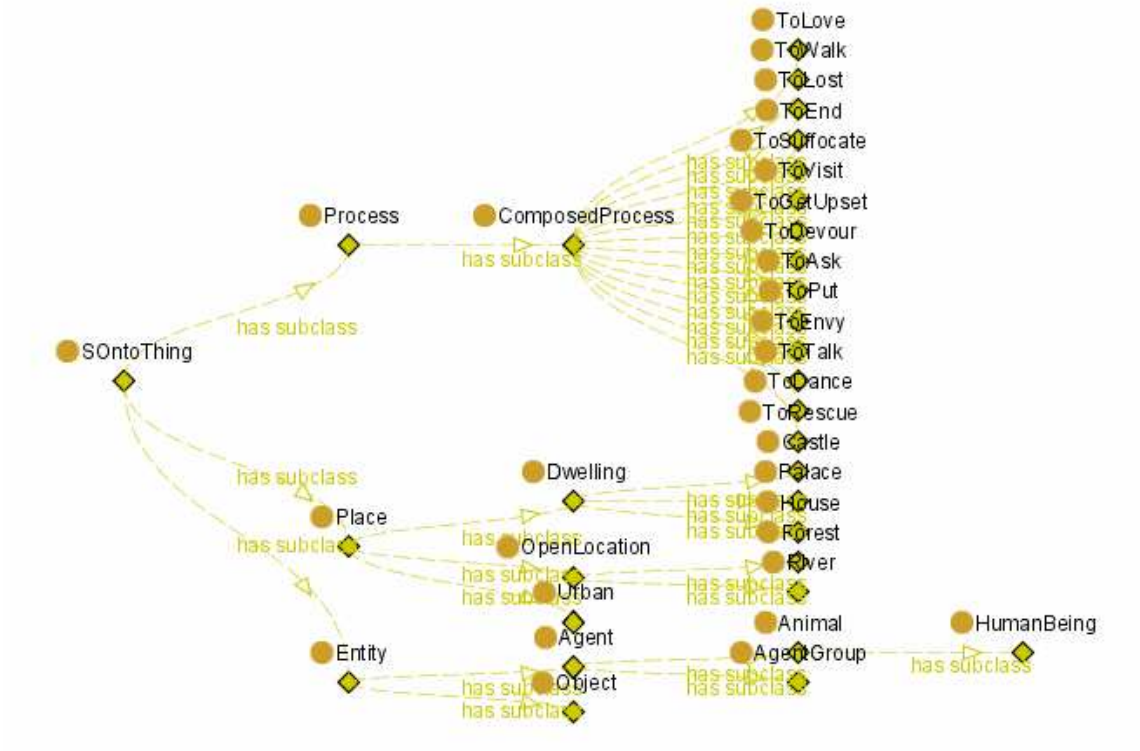


Figura 3.8: Gráfica de las partes de un cuento

Propiedades de los cuentos

Con lo anterior se pueden definir todos los conceptos necesarios para los elementos que forman el cuento. Sin embargo sólo con eso no sería suficiente. Es necesario definir propiedades para que esos conceptos estén mejor definidos y se enriquezca la representación de los cuentos. Estas propiedades son fundamentales para que todos los conceptos tengan sentido y estén relacionados los unos con los otros, para que entre todos formen una historia concreta y a la hora de construir otro cuento estas propiedades se tengan en cuenta para aumentar la coherencia del guión.

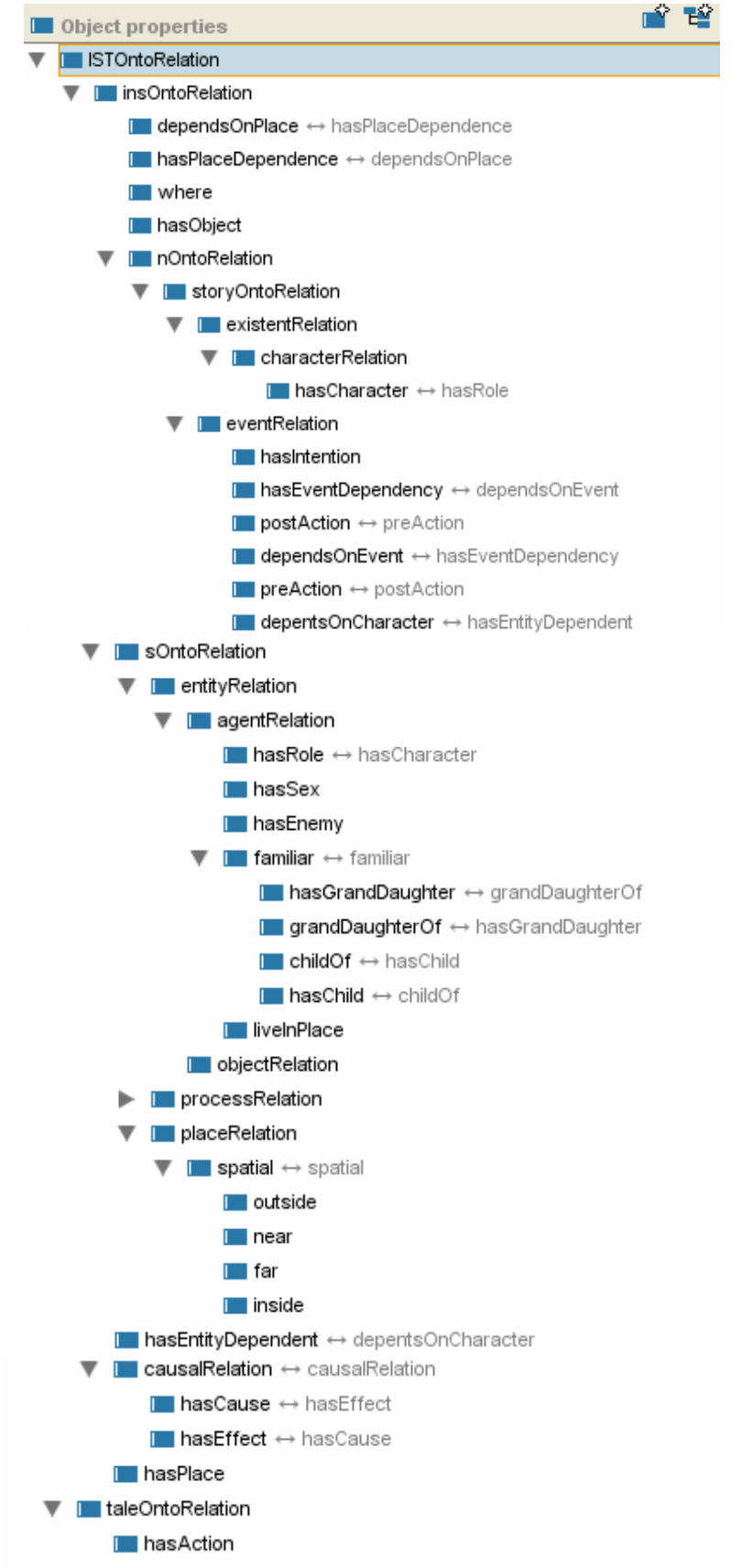


Figura 3.9: Visión general de las propiedades de la ontología

En la actualidad hemos considerado más propiedades de las que están siendo utilizadas, pero que en un futuro sí podrían ser necesarias, para enriquecer más la representación de un cuento.

Dentro de todas las propiedades definidas, aquellas que consideramos las más importantes para nuestro sistema son:

- *where*: para especificar el lugar de una acción
- *hasObject*: para determinar la implicación de algún objeto de los definidos dentro de SOntoThing-Entity-Object
- *hasRole*: para determinar el personaje que desempeña un rol concreto dentro del cuento.
- *postAction* y *preAction*: para especificar la secuencia de acciones.
- *DependsOnCharacter*: para determinar los personajes que están implicados en una acción.
- *hasAction*: para poder determinar las acciones de los cuentos.
- *hasCause* y *hasEffect*: especifica las acciones definidas como no de Propp que son consideradas causa o efecto de las de Propp.

3.1.2 Ejemplo de representación de un cuento

Para mostrar de un modo más claro la estructura de nuestra ontología, y cómo definir un cuento, veremos la aplicación práctica de representar un cuento con nuestra ontología.

El cuento de Caperucita Roja

Para representar un cuento lo primero que debemos hacer es identificar los siguientes elementos dentro del mismo:

- Personajes que aparecen, identificando su papel en la historia (principal, secundario…)
- Lugares importantes del cuento.
- Objetos que influyen de un modo u otro en el transcurso del cuento.
- Acciones del cuento, correspondientes a las funciones de Propp y acciones que también sería importante definir pero que no se corresponden con ninguna función de Propp.

A continuación rellenamos la parte del cuento común a la estructura definida por las funciones de Propp. Para ello creamos una instancia de cada una de las funciones identificada en el cuento y la nombramos, precedida de la inicial del cuento para poder saber a qué cuento pertenece. Es importante tener en cuenta que en ocasiones, una función de Propp puede tener subclases más específicas, que definen con más precisión la acción del cuento que queremos representar.

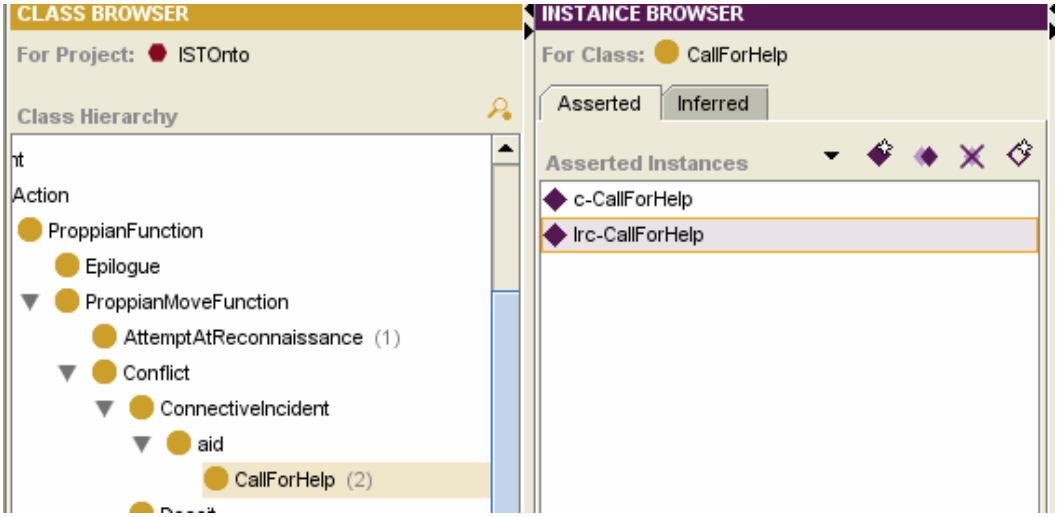


Figura 3.10: Acción del cuento de caperucita y su instancia.

Para cada una de estas acciones es necesario rellenar las correspondientes propiedades que han sido definidas en su rango de actuación. Las propiedades más importantes son las que nos indican el lugar, los objetos y personajes implicados en la misma, así como la acción anterior y posterior dentro del cuento, estas nos sirven para poder seguir un orden en el guión del cuento.
Siguiendo con el ejemplo de caperucita, en la figura podemos observar cómo se han rellenado las propiedades de la acción *HeroDeparture*, que corresponde con la partida de Caperucita a casa de su abuela. Como *preAction* (acción anterior) no tiene ninguna, ya que es el principio del cuento, en cambio observamos que sí tiene acción posterior. La propiedad *dependsOnCharacter* contiene al personaje que aparece en la acción, en este caso Caperucita y como lugar aparece la casa de la madre.

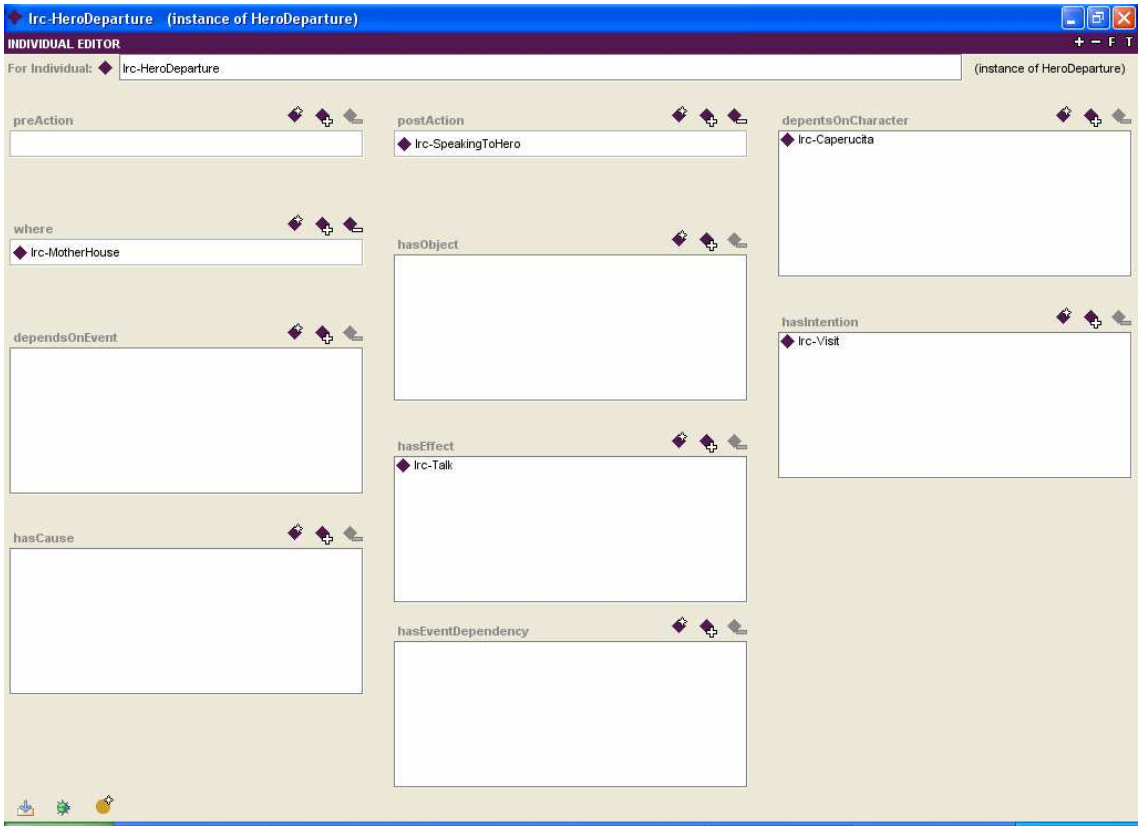


Figura 3.11: Representación de una acción del cuento.

Una vez definidas todas las acciones, podemos definir los personajes del cuento. Debemos definir los roles que desempeñan los personajes en el cuento, y crear una instancia por cada uno, como en el caso anterior.

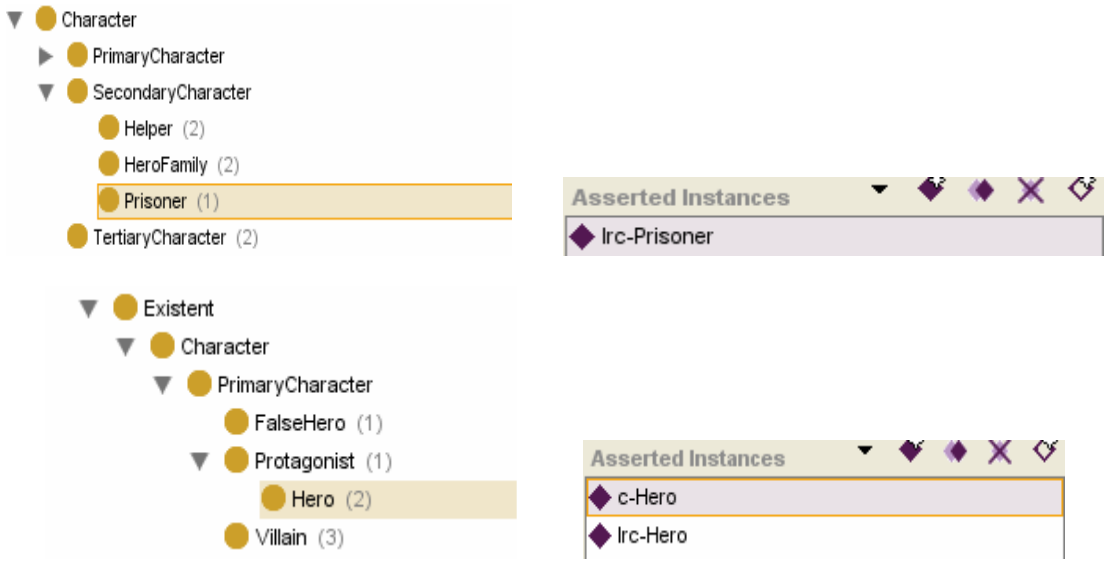


Figura 3.12: Definición del rol de un personaje.

Los lugares y los objetos debemos definirlos como individuos del concepto correspondiente. Por ejemplo, en el cuento de Caperucita Roja, definiremos el bosque como individuo del concepto Forest, y la cesta como individuo de Object. Al igual que con las acciones, los conceptos generales que modelan lugares y objetos, tiene subconceptos que aproximan más los individuos con los que nos podemos encontrar.

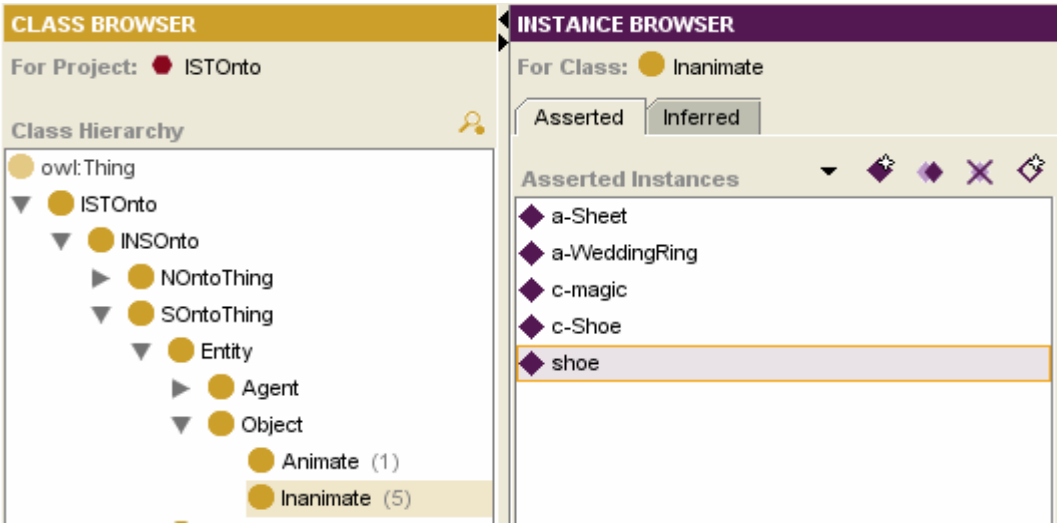


Figura 3.13: Conceptos e individuos correspondientes a los objetos.

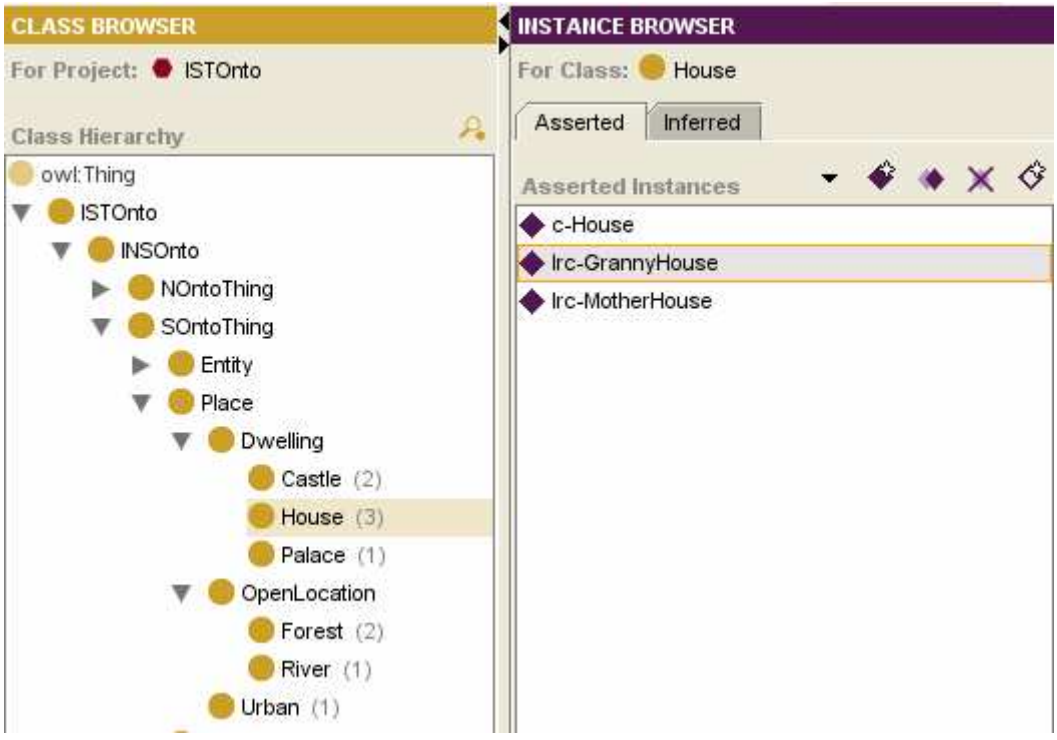


Figura 3.14: Conceptos e individuos correspondientes a los lugares.

A continuación, podemos definir algunas acciones que consideremos importantes en nuestro cuento que no se correspondan con las funciones de Propp, son las que hemos definido bajo el concepto Process.

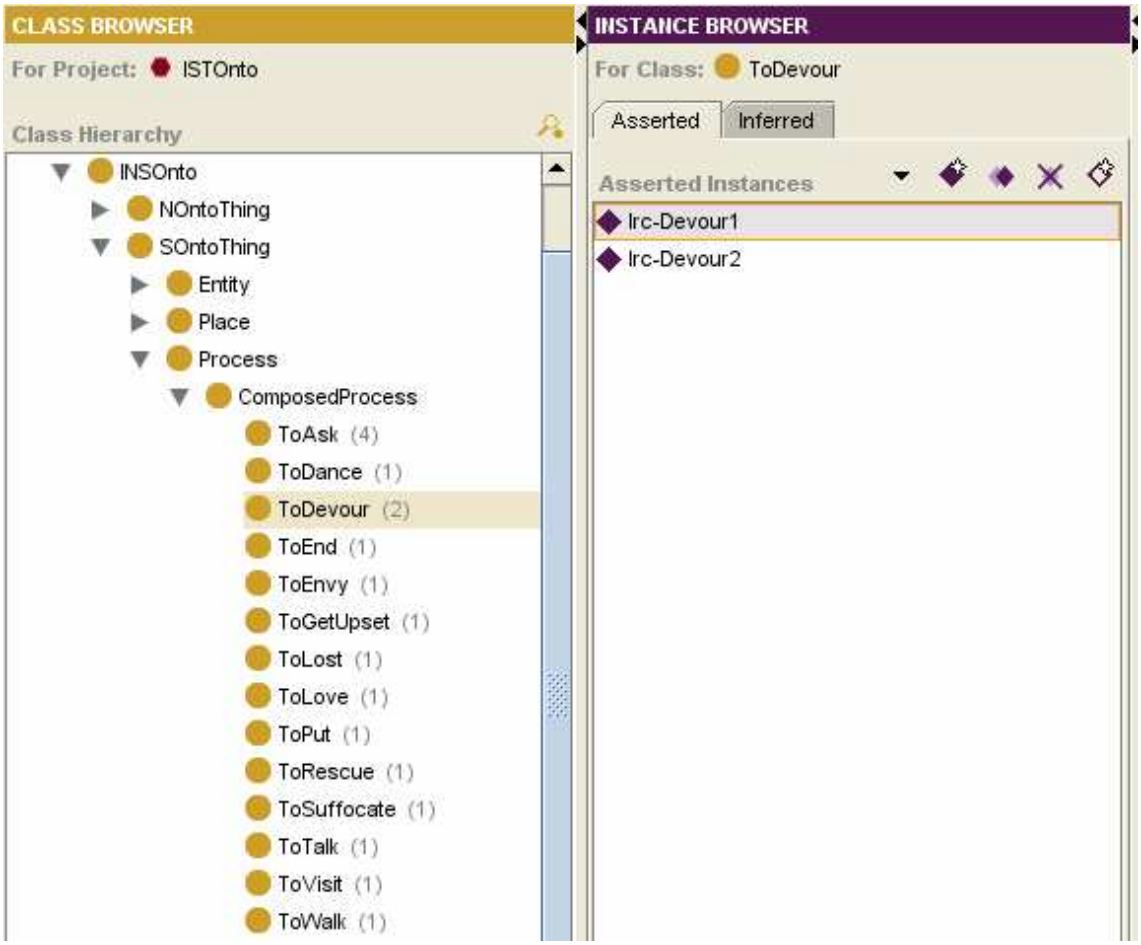


Figura 3.15: División de otras acciones (procesos) que ocurren en algunos cuentos y que no se corresponden con ninguna función de Propp.

3.2 Base de Casos

La segunda subclase de ISTOnto, clase principal de la ontología, es la clase TALEOnto. En esta subclase es donde se definen los casos que son la base del sistema de razonamiento. Las subclases de TALEOnto son todos los casos que tiene nuestro sistema. Cada caso tiene definido el guión de cada uno de los cuentos representados en la ontología. Cada uno de ellos tiene una instancia de su propio concepto.

Figura 3.16: Conceptos correspondientes al guión de cada historia.

Una de las propiedades definida es *hasAction*. Esta propiedad tiene gran importancia en la base de casos ya cada una de las instancias que hay en esta parte va a tener en los valores de dicha propiedad todas las acciones que forman el guión del cuento que representa.

Vamos a explicar ahora cómo añadir un nuevo caso a la base. Para ello vamos a recordar los conceptos y propiedades que componen el cuento de Caperucita:

Personajes del cuento:

lrc-Caperucita

Instancia de HumanBeing

Propiedades:

hasRol: lrc-Hero, *hasSex*: woman,
liveInPlace: lrc-MotherHouse

lrc-MadreCaperucita

Instancia de HumanBeing

Propiedades:

hasRol: lrc-HeroFamily, *hasSex*: woman,
liveInPlace: lrc-MotherHouse

lrc-AbuelaCaperucita

Instancia de HumanBeing

Propiedades:

hasRol: lrc-Prisoner, *hasSex*: woman,
liveInPlace: lrc-GrannyHouse

lrc-Wolf

Instancia de Animal

Propiedades:

hasRol: lrc-Villain, *hasSex*: man,
liveInPlace: lrc-Forest

lrc-Leñadores

Instancia de AgentGroup

Propiedades:

hasRol: lrc-Helper, *hasSex*: man

Figura 3.17: Detalle de los personajes del cuento de Caperucita

Lugares del cuento

lrc-MotherHouse

Instancia de House

lrc-GrannyHouse

Instancia de House

lrc-Forest

Instancia de Forest

lrc-River

Instancia de River

Figura 3.18: Detalle de los lugares del cuento de Caperucita

Objetos:

lrc-Cestita

Instancia de Inanimate

lrc-Piedras

Instancia de Inanimate

Figura 3.19: Detalle de los objetos del cuento de Caperucita

Acciones

lrc-HeroDeparture

Instancia de HeroDeparture

Propiedades:

preAction: (vacia), *postAction:* lrc-SpeakingToHero,
dependsOnCharacter: lrc-Caperucita,
where: lrc-MotherHouse, *hasObject:* lrc-Cestita

lrc-SpeakingToHero

Instancia de SpeakingToHero

Propiedades:

preAction: lrc-HeroDeparture,
postAction: lrc-VillainKidnappingHeroFamily,
dependsOnCharacter: lrc-Caperucita, lrc-Wolf
where: lrc-Forest

lrc-VillainKidnappingHeroFamily

Instancia de VillainKidnappingHeroFamily

Propiedades:

preAction: lrc-SpeakingToHero,
postAction: lrc-FalseSubstitution,
dependsOnCharacter: lrc-AbuelaCaperucita, lrc-Wolf
where: lrc-GrannyHouse

lrc-FalseSubstitution

Instancia de FalseSubstitution

Propiedades:

preAction: lrc- VillainKidnappingHeroFamily,
postAction: lrc-CallForHelp,
dependsOnCharacter: lrc-Wolf
where: lrc-GrannyHouse

lrc-CallForHelp

Instancia de CallForHelp

Propiedades:

preAction: lrc- FalseSubstitution,
postAction: lrc-VillainAttemptToDevourHero,
dependsOnCharacter: lrc-Leñadores
where: lrc-Forest

lrc- VillainAttemptToDevourHero

Instancia de VillainAttemptToDevourHero

Propiedades:

preAction: lrc- CallForHelp,
postAction: lrc-ReleaseFromCpativity,
dependsOnCharacter: lrc-Caperucita, lrc-Wolf
where: lrc-GrannyHouse

lrc-ReleaseFromCaptivity

Instancia de ReleaseFromCaptivity

Propiedades:

preAction: lrc- VillainAttemptToDevourHero,
postAction: lrc-Punishment,
dependsOnCharacter: lrc-Caperucita, lrc-AbuelaCaperucita, lrc-Leñadores
where: lrc-GrannyHouse

lrc-Punishment

Instancia de Punishment

Propiedades:

preAction: lrc- ReleaseFromCaptivity,
postAction: (vacia),
dependsOnCharacter: lrc-wolf, lrc-Leñadores
where: lrc-River, hasObject: lrc-Piedras

Figura 3.20: Detalle de las Acciones que componen el cuento de Caperucita.

.

Una vez que todo lo anterior está definido podemos añadir un nuevo caso a la base de casos, es decir una subclase a la clase TALEOnto. Esta subclase se llamará como el titulo del cuento, Caperucita, y tendrá una instancia, de nombre TaleCape, que representa el caso en sí.

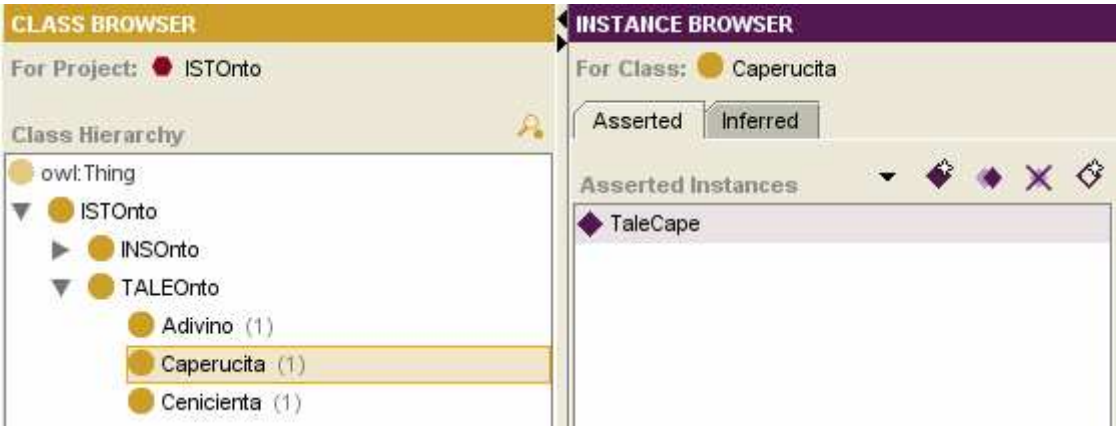


Figura 3.21: Concepto que representa al cuento junto con su instancia.

De este último individuo tendremos que rellenar la propiedad *hasAction*, que es la que relaciona todas las acciones del cuento, con el guión del mismo. Así tenemos que cada cuento queda representado por las acciones que lo componen y cada una de ellas por otras propiedades tales como personajes implicados en ella, lugar donde se desarrollan, etc.

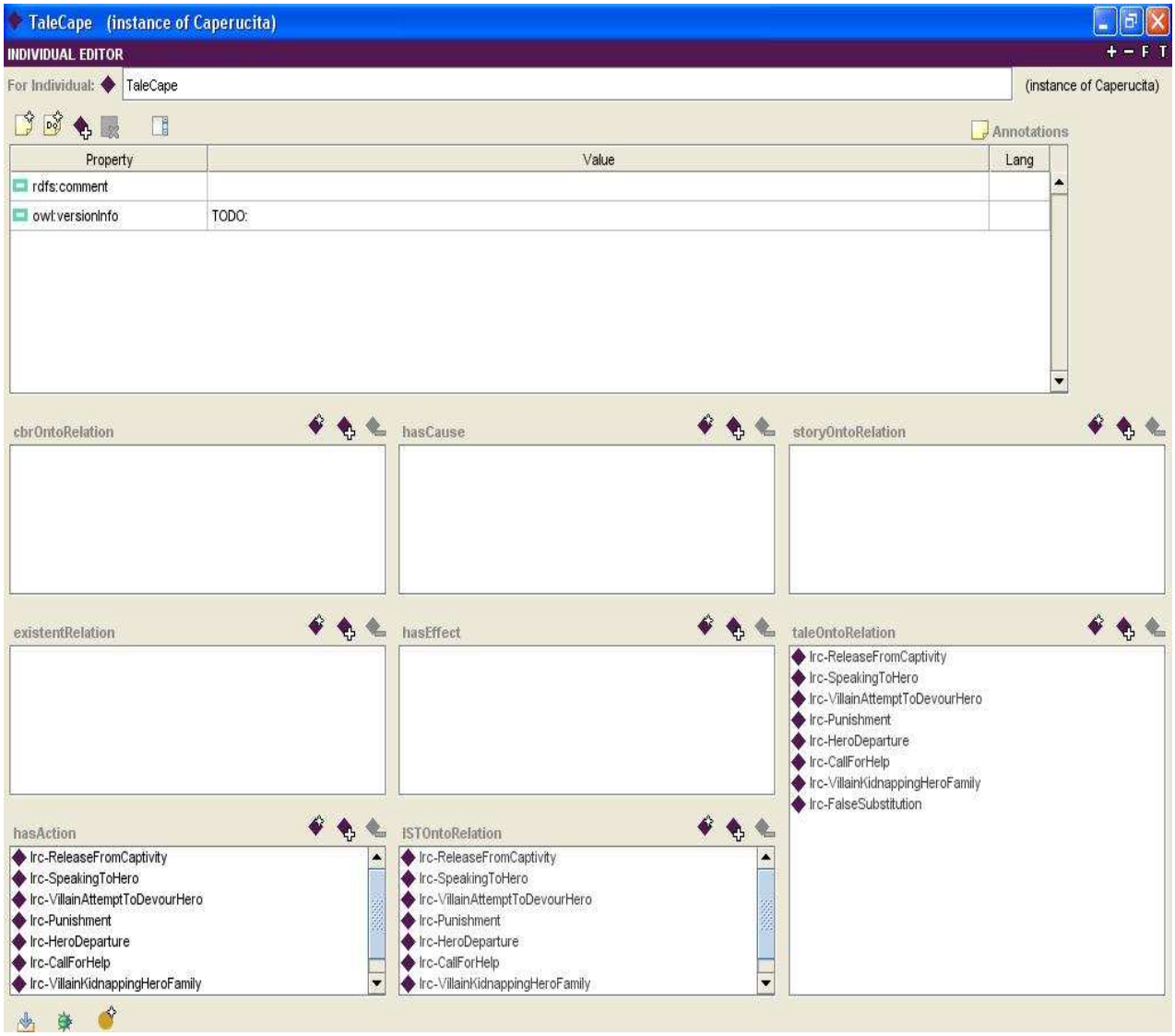


Figura 3.22: Representación del guión del cuento de Caperucita Roja.

Este ejemplo muestra cómo se representa un cuento en nuestro sistema. En la siguiente figura podemos observar el guión completo de un cuento a partir de sus individuos. La instancia del cuento TaleCape tiene como acciones las instancias unidas a través de flechas de color morado. A su vez, estas se relacionan con su pre-acción y su post-acción mediante las flechas de color rojo y azul respectivamente. Las flechas verdes relacionan las acciones con el lugar donde se desarrollan, por último los personajes involucrados en una acción están determinados a través de las flechas amarillas y naranjas.

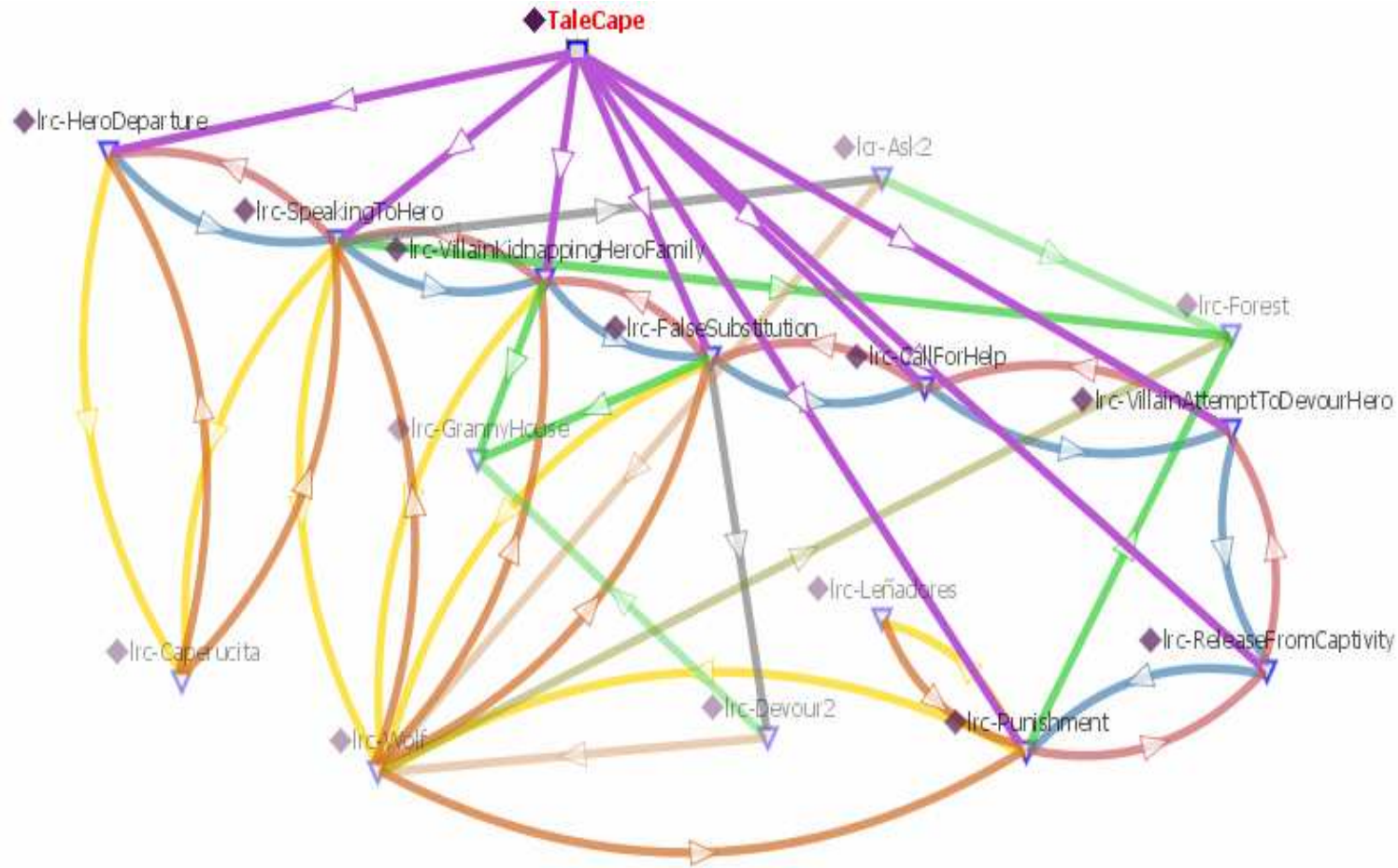


Figura 3.23: Guión del cuento de Caperucita

De esta manera se definen todos los casos de nuestro proyecto. La base de casos de nuestro sistema está compuesta por los siguientes cuentos:

- Caperucita Roja
- Cenicienta
- El Adivino
- El Pez de Oro
- La zorra, la liebre y el gallo.
- The Magic Swan-Geese
- El Patito feo

La mayoría de ellos están compuestos por una media de unas diez acciones, cinco personajes, tres lugares y uno o dos objetos. Sin embargo, está base de casos se va ampliando gracias al aprendizaje del sistema y al editor de cuentos que hemos creado para facilitar la adquisición de casos y que explicaremos en el siguiente apartado.

3.3 Adquisición de conocimiento: EDITOR DE CUENTOS

Como vimos en los objetivos del proyecto, la adquisición de casos que enriquecieran la base de conocimiento supuso un gran cuello de botella en nuestro sistema. Para poder generar cuentos originales, creativos y coherentes era muy necesario que la base de casos fuera rica y abundante, pero conseguir estas dos características sería muy costoso. Por eso, nos planteamos la posibilidad de crear un editor de cuentos para poder facilitar la adquisición de conocimiento. La idea era crear una interfaz amigable para poder introducir datos en nuestra base de conocimiento de manera fácil y rápida. El editor de cuentos responde a estas necesidades. El editor está pensado para que cualquier usuario, sin experiencia en el manejo de ontologías, pueda crear un cuento que enriquezca la base de conocimiento de nuestro sistema, ya que esta interfaz a través de Ontobridge transforma todos los datos introducidos relativos al cuento a un fichero OWL.

3.3.1 Funcionamiento del Sistema

Cada vez que el usuario introduce cualquier elemento de los que componen un cuento y que se definen en el editor, estos se almacenan en una estructura de datos, llamada *TransferTale*, que podemos observar en la figura 3.24 y que está compuesta por diferentes listas para las estructuras en las que se definen los personajes, los lugares, los objetos y las acciones, todos ellos con sus correspondientes propiedades.

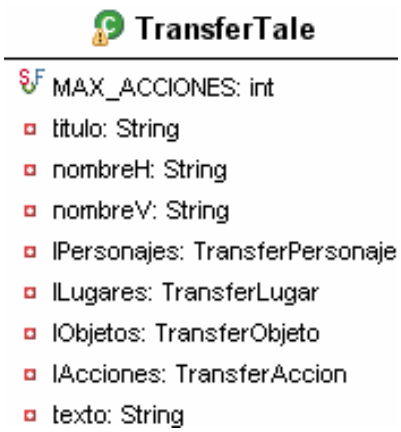


Figura 3.24: Estructura de datos *TransferTale*

Al pulsar el botón Aceptar del formulario principal del editor es cuando a través de las funciones de OntoBridge se almacena el cuento en la base del conocimiento. Hasta que el usuario no pulsa aceptar no se guardan los cambios en la base de conocimiento ya que de otra manera podrían crearse cuentos incompletos o que hagan inconsistente la ontología. Aunque hay que mencionar que tiene el inconveniente de que si ocurriese algún tipo de problema durante la creación del cuento, finalizando la aplicación de forma repentina, los elementos introducidos se perderían y habría que introducir el cuento de cero.

El proceso de creación es el mismo que el que se explicó en la sección 3.3.4, cuando se expuso un ejemplo de cómo crear un cuento utilizando el programa Protégé. El cuento nuevo será identificado en la ontología con el nombre del título elegido por el usuario precedido por TALE. Todas las instancias que pertenezcan al cuento estarán precedidas por las tres primeras letras del título del cuento seguidas por el nombre que les dio el usuario. De todo esto se encarga directamente el sistema de manera que el usuario no tiene que preocuparse por ello y permite que la representación de todos los cuentos sea igual.

3.3.2 Interfaz de usuario

Como ya hemos comentado, el objetivo del editor de cuentos era crear una interfaz sencilla y cómoda de manejar para poder introducir datos en nuestra base de conocimiento de forma rápida. A continuación explicaremos cómo es esta interfaz, qué componentes forman parte de ella y cómo se usa.

Como pantalla principal de nuestro editor nos encontramos con un formulario en el que tendremos que introducir distinta información sobre el cuento. En la parte superior de la pantalla encontramos un espacio donde poder escribir el título del cuento que vamos a editar. En el centro de la pantalla encontramos una serie de botones que al pincharlos desplegarán otras pantallas en las que podremos crear distintos ingredientes de nuestro cuento. Por último, en el recuadro inferior se visualizará la información que vamos introduciendo, relativa a nuestro cuento.

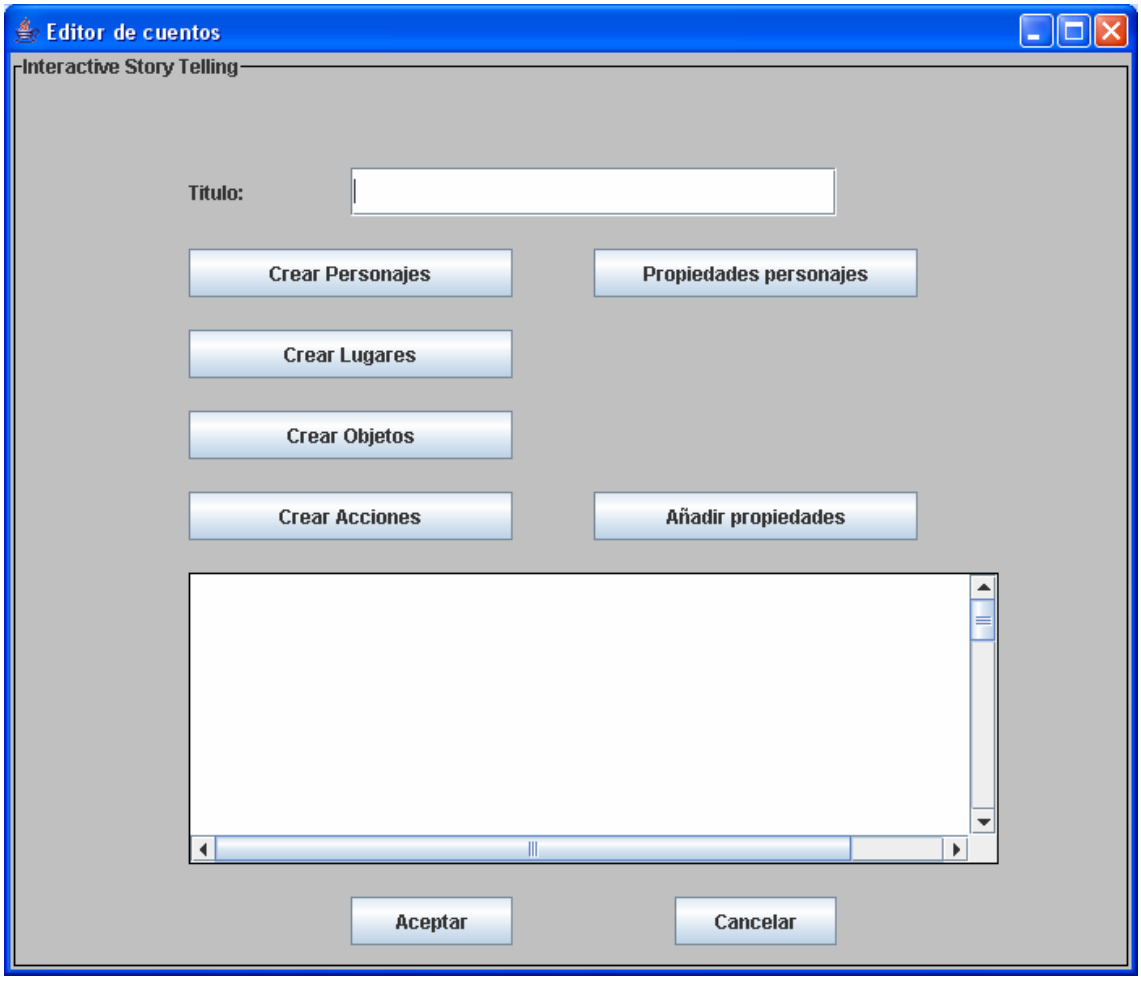


Figura 3.25: Pantalla principal del Editor de cuentos.

Mediante los distintos botones asociados a cada componente de un cuento podemos ir creando la información relativa a cada uno de estos componentes. Es decir, al pulsar sobre el botón de *Crear personaje*, nos encontraremos con un nuevo formulario a rellenar con la información relativa a un personaje, y así con cada uno de los componentes.

Los formularios de personajes, lugares, objetos y acciones, solicitan información sobre el nombre que le vamos a dar a cada cosa y sobre el tipo de personaje, lugar, objeto o acción es.

Sobre los personajes del cuento necesitamos la siguiente información: el nombre del personaje, el papel que desempeña en el cuento (los tipos de papeles que podemos encontrar también se encuentran definidos en la morfología de los cuentos de Propp), y por último el tipo de personaje (agente, grupo de agentes, animal o ser humano)

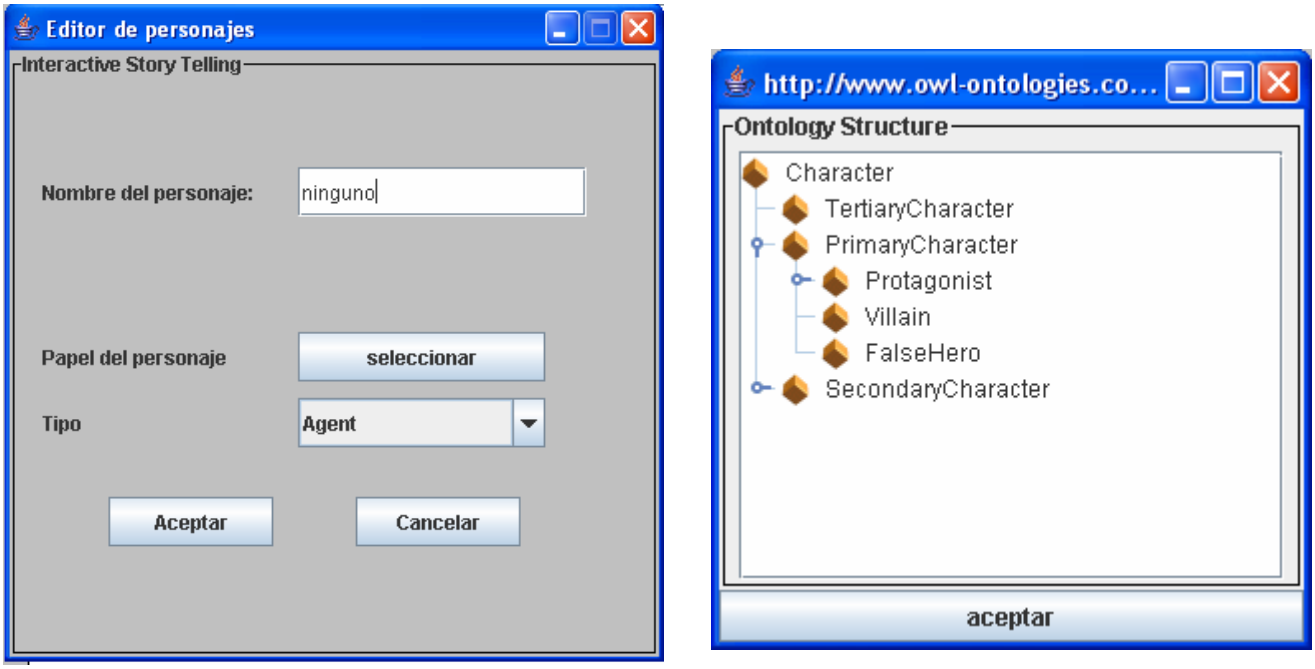


Figura 3.26: Formularios para la creación de un personaje.

Sobre los lugares sólo nos interesa saber su nombre y qué tipo de lugar es, un bosque, un castillo, una casa, etc…

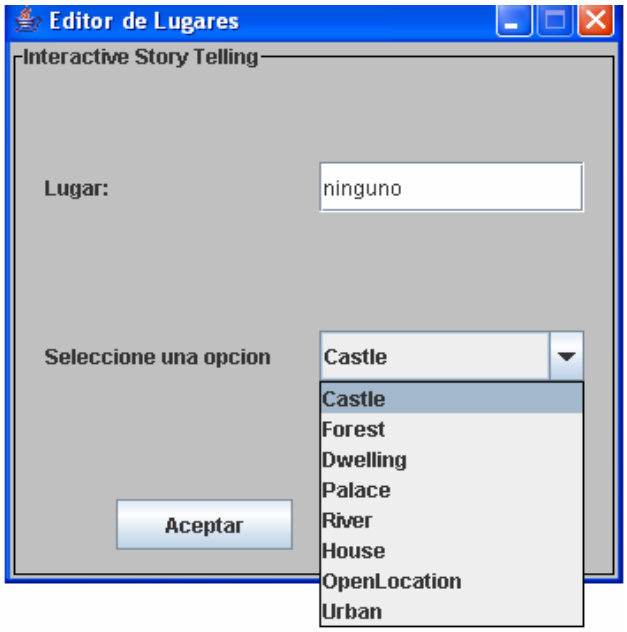


Figura 3.27: Interfaz para la creación de un lugar.

De los objetos que aparecen en el cuento, necesitamos conocer aquellos que de una forma u otra influyen en el transcurso del guión. Objetos necesarios para realizar una acción, objetos recibidos como regalos, etc… De estos objetos necesitamos saber su nombre, y si son animados o inanimados.

Figura 3.28: Interfaz de creación de un nuevo objeto.

Para las acciones necesitamos saber su nombre (aunque simplemente lo utilizamos como ayudar para el usuario, ya que después en la ontología el nombre que le hayamos asignado no se utiliza) y la función de Propp con la que se corresponde. Es posible que en muchas ocasiones no sepamos con certeza a qué función de Propp puede corresponder una determinada acción, pero en general, la asociamos con aquella que nos parece pueda tener más sentido.

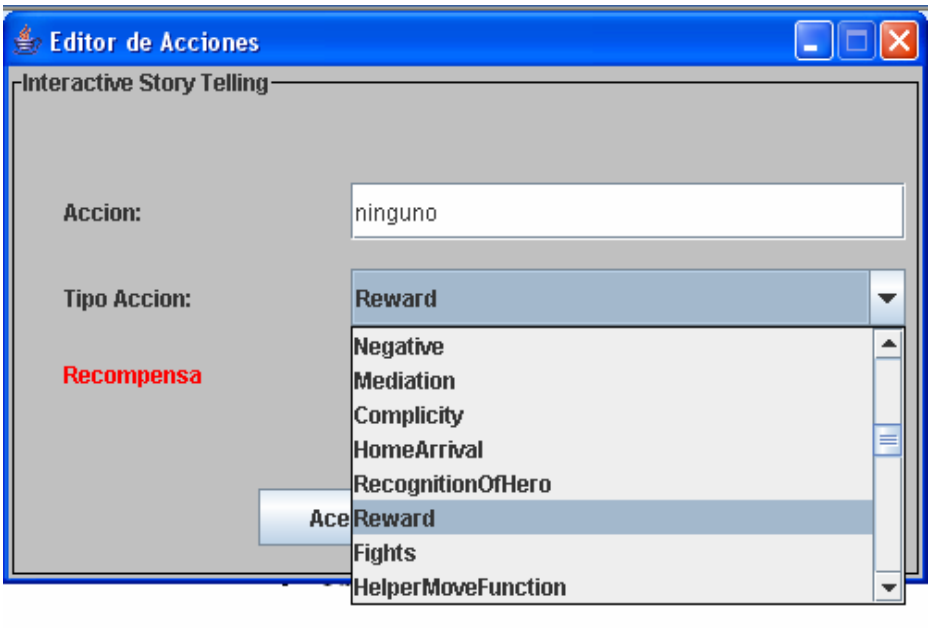


Figura 3.29: Interfaz de creación de una nueva acción.

Después de que ya hemos creado todos los ingredientes de nuestro cuento, pasamos a relacionarlos entre sí, para que el cuento tenga sentido. Esto lo conseguimos a través de los formularios de *Crear propiedades personajes* y *Crear propiedades acciones.*

Comenzaremos introduciendo las propiedades de un personaje, que se corresponden con el lugar en el que vive y el sexo al que pertenece. Con los botones anterior y siguiente, recorreremos todos los personajes.

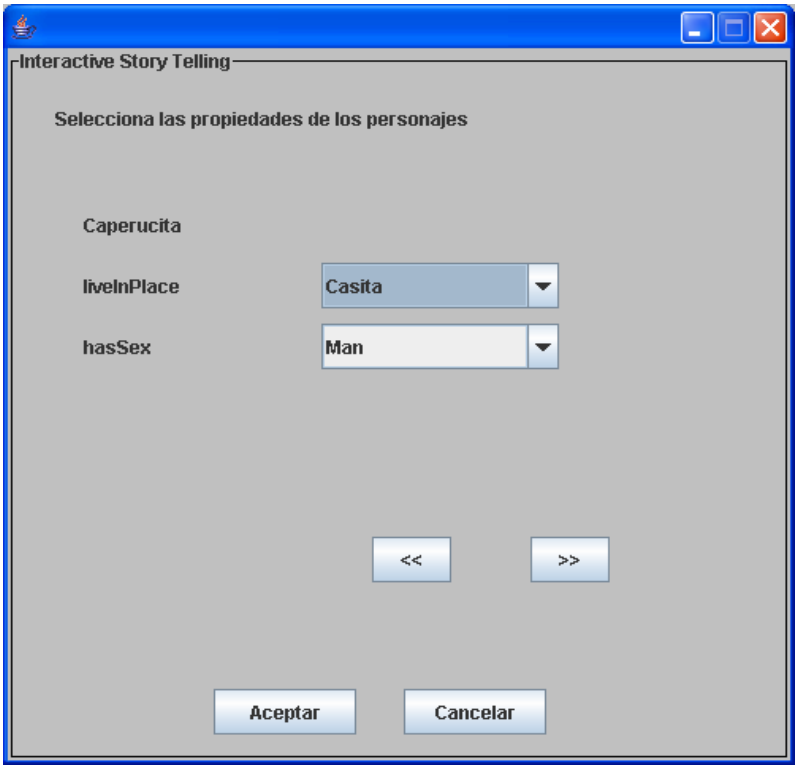


Figura 3.30: Formulario para añadir las características de los distintos personajes

Por último debemos completar las propiedades de una acción. Esas propiedades que permiten enlazar por completo el cuento que hemos creado. Ya que una acción compromete a los personajes involucrados en ella, al lugar en el que se desarrolla y al objeto que necesita, en el caso de necesitar alguno.
A través del último formulario podemos completar toda esta información relativa a las acciones de nuestro cuento.

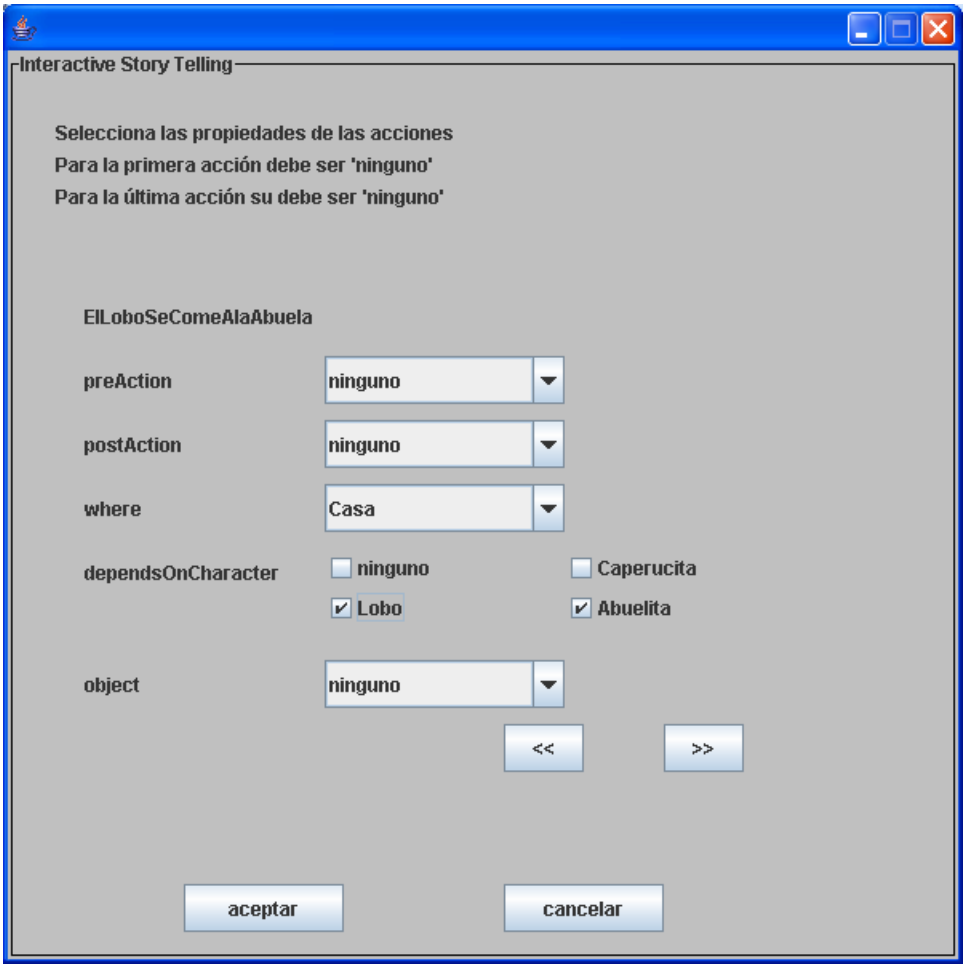


Figura 3.31: Formulario correspondiente a las propiedades de las acciones.

Una vez que ya tenemos completa toda la información de nuestro cuento, procedemos a crearlo pulsando el botón Aceptar en la pantalla principal, y si todo ha salido bien, habremos conseguido añadir un cuento nuevo a nuestra base de conocimiento.

4. GENERADOR DE NARRACIÓN INTERACTIVA

Una vez que decidimos la estructura de la ontología, cómo serían representados los cuentos, y que implementamos el editor de manera que pudiéramos poblar la base del conocimiento más fácilmente, pasamos a implementar el generador de cuentos.

Por un lado deberíamos tomar ciertas decisiones en torno a los datos de entrada al sistema, es decir, qué nivel de implicación le concederíamos a nuestro usuario en la creación de la historia. En función de este nivel los datos de entrada al generador variarían. Por otro lado, debíamos construir un módulo de razonamiento que fuera creativo y que además permitiera la interacción con el usuario. De estos objetivos nacieron una interfaz de usuario sencilla que permitiera que el usuario pudiera elegir determinadas características de su cuento, pero a la vez dejando libertad al generador, y un sistema de razonamiento basado en casos empleando la técnica de Adaptación Constructiva, que explicaremos más adelante.

Además para observar las diferencias que se obtienen utilizando métodos de razonamiento transformativos frente a los métodos generativos realizamos una segunda implementación del sistema de razonamiento basado en el método transformativo.

4.1 Interfaz de usuario

Para que el sistema pueda generar un cuento necesita unos datos iniciales. Estos datos se recogen a través de una serie de formularios. Como acabamos de comentar decidimos que como datos de entrada el sistema preguntaría al usuario por dos personajes que quiere que aparezcan en su cuento. La elección de estos dos personajes se realiza en base a los que ya existen en la base de conocimiento del sistema, pero el usuario podrá modificar el nombre de dichos personajes. El que elija dos personajes que sean iguales a unos que ya existen quiere decir que su personaje tendrá las mismas características del personaje elegido. Después tendrá que elegir un lugar donde quiere que se desarrolle una parte de la historia. Además deberá elegir un objeto que será necesario para el buen transcurso del guión de cuento, y por último deberá elegir tres tipos de acciones que quiere que sucedan en su cuento. Estas acciones se corresponden con las ya mencionadas funciones de Propp.

Pantalla principal

La interfaz del generador está compuesta por un formulario principal con varias pestañas. Cada una de estas pestañas se corresponde con distintos formularios en los que podrá seleccionar los diferentes datos de entrada al generador.
En la primera de ellas podemos encontrar unas breves instrucciones del funcionamiento del generador.

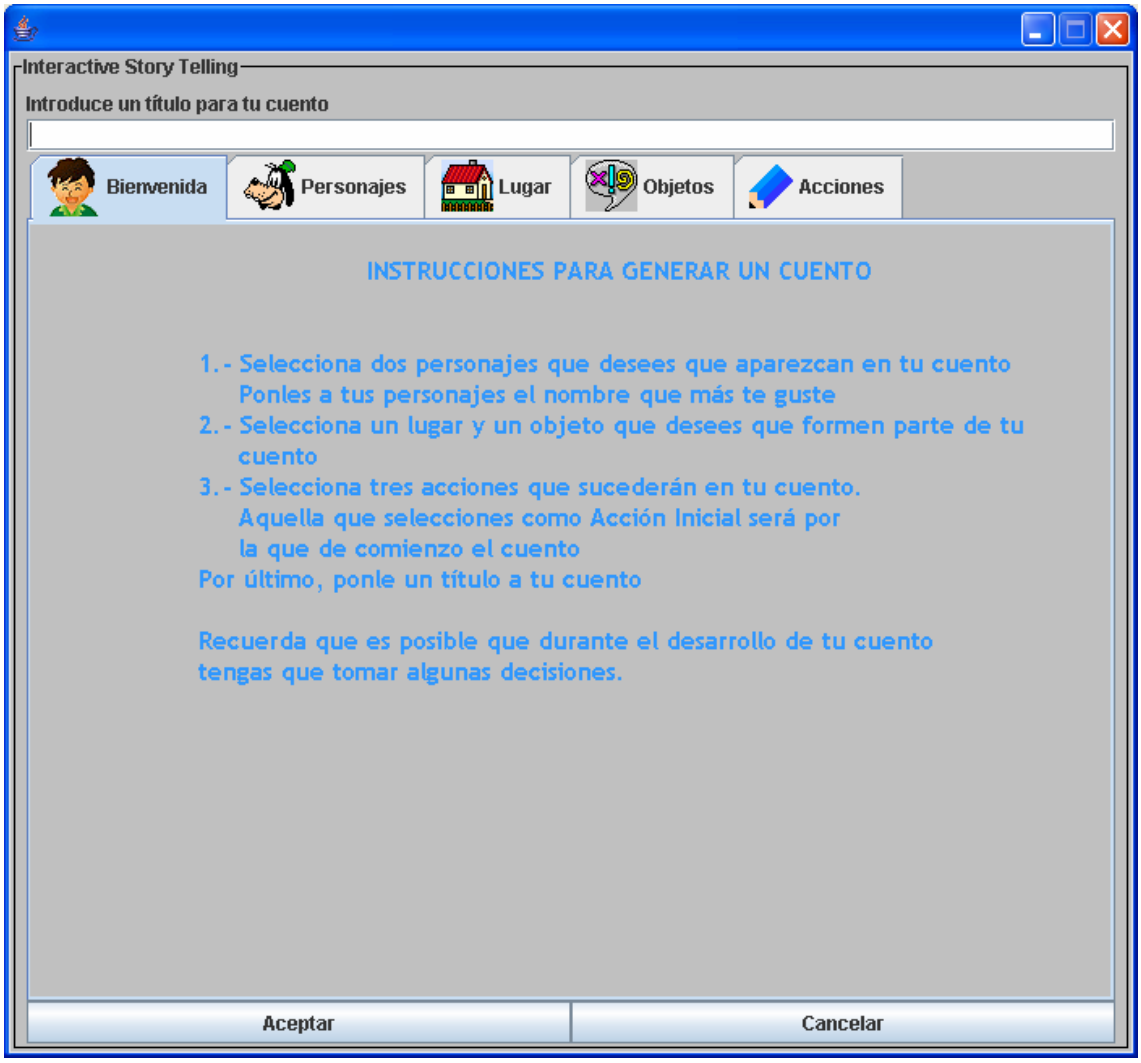


Figura 4.1: Instrucciones del generador

Elección de personajes

Como acabamos de comentar decidimos que como datos de entrada el sistema preguntaría al usuario por dos personajes que quiere que aparezcan en su cuento. Su elección se realiza en base a los que ya existen en la base de conocimiento del sistema. Para que aparezcan en la ventana del formulario deberemos pasar el ratón por encima del centro del formulario (esto habrá que hacerlo en todas las pestañas que componen el generador). Aparecerán dos listas desplegables con todos los personajes que podemos elegir. Podremos modificar el nombre de los personajes elegidos pulsando en los botones que se encuentran al lado de las listas. El que elijamos dos personajes que sean iguales a unos que ya existen quiere decir que su personaje tendrá las mismas características del personaje elegido.

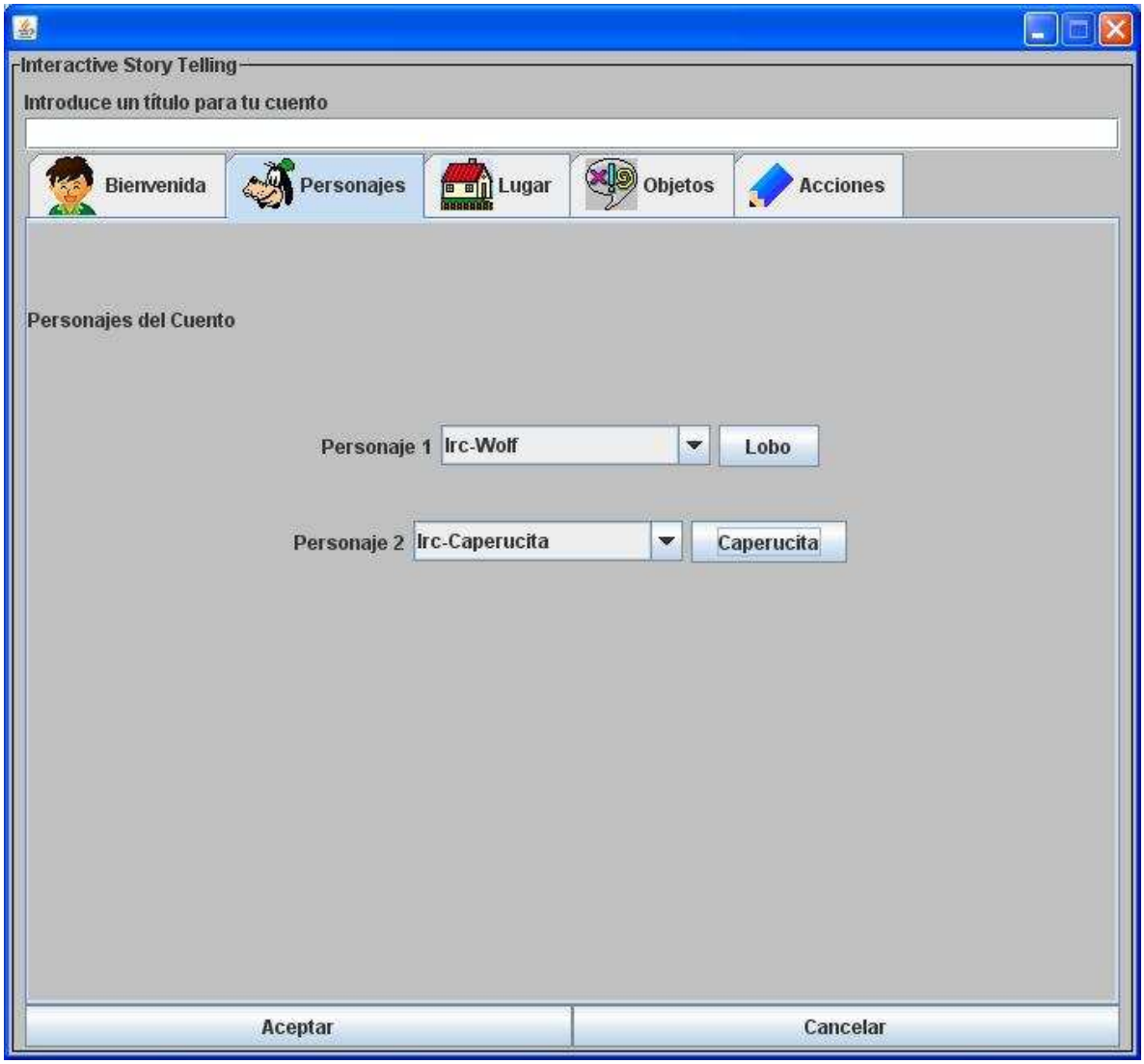


Figura 4.2: pestaña de elección de personajes.

Elección de Lugar

Después tendrá que elegir un lugar donde quiere que se desarrolle una parte de la historia. Podemos elegir entre los tipos de lugares que se encuentran en la base de conocimiento.

Figura 4.3: Pestaña de elección del lugar del cuento.

Elección de objeto

Debemos elegir un objeto que queremos que aparezca en nuestro cuento. Este objeto es posible que sea necesario para realizar alguna acción, o que se corresponda con algún regalo que aparezca en la historia... eso dependerá de nuestro cuento.

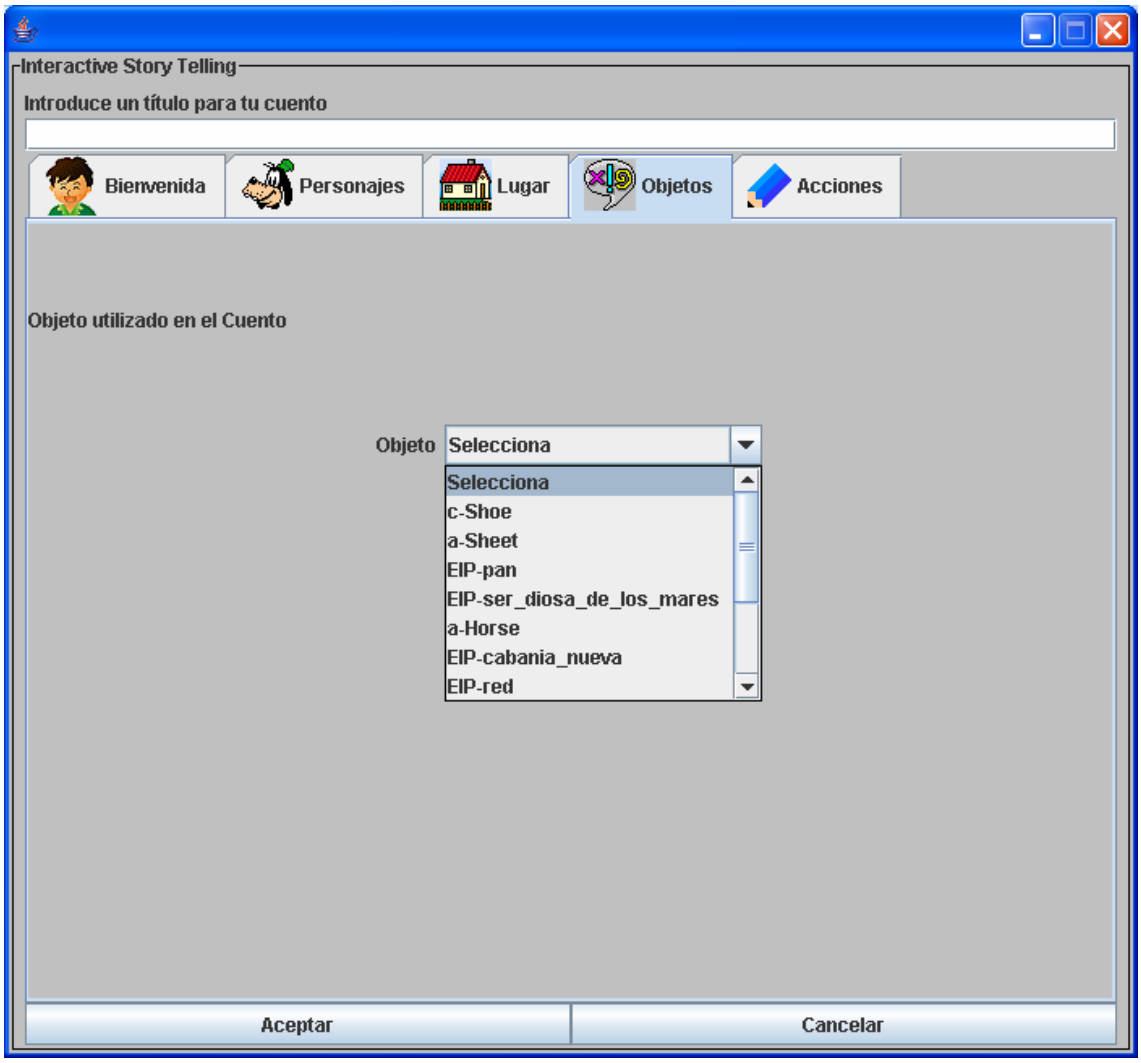


Figura 4.4: Pestaña de elección del objeto.

Elección de acciones

Por último debemos elegir tres tipos de acciones que queremos que sucedan en nuestro cuento. Estas acciones se corresponden con las ya mencionadas funciones de Propp.
La acción elegida como *Acción Inicial* será aquella por la que empezará nuestro cuento. Por ejemplo, si elegimos *Conflict* nuestro cuento comenzará con un conflicto entre varios personajes.

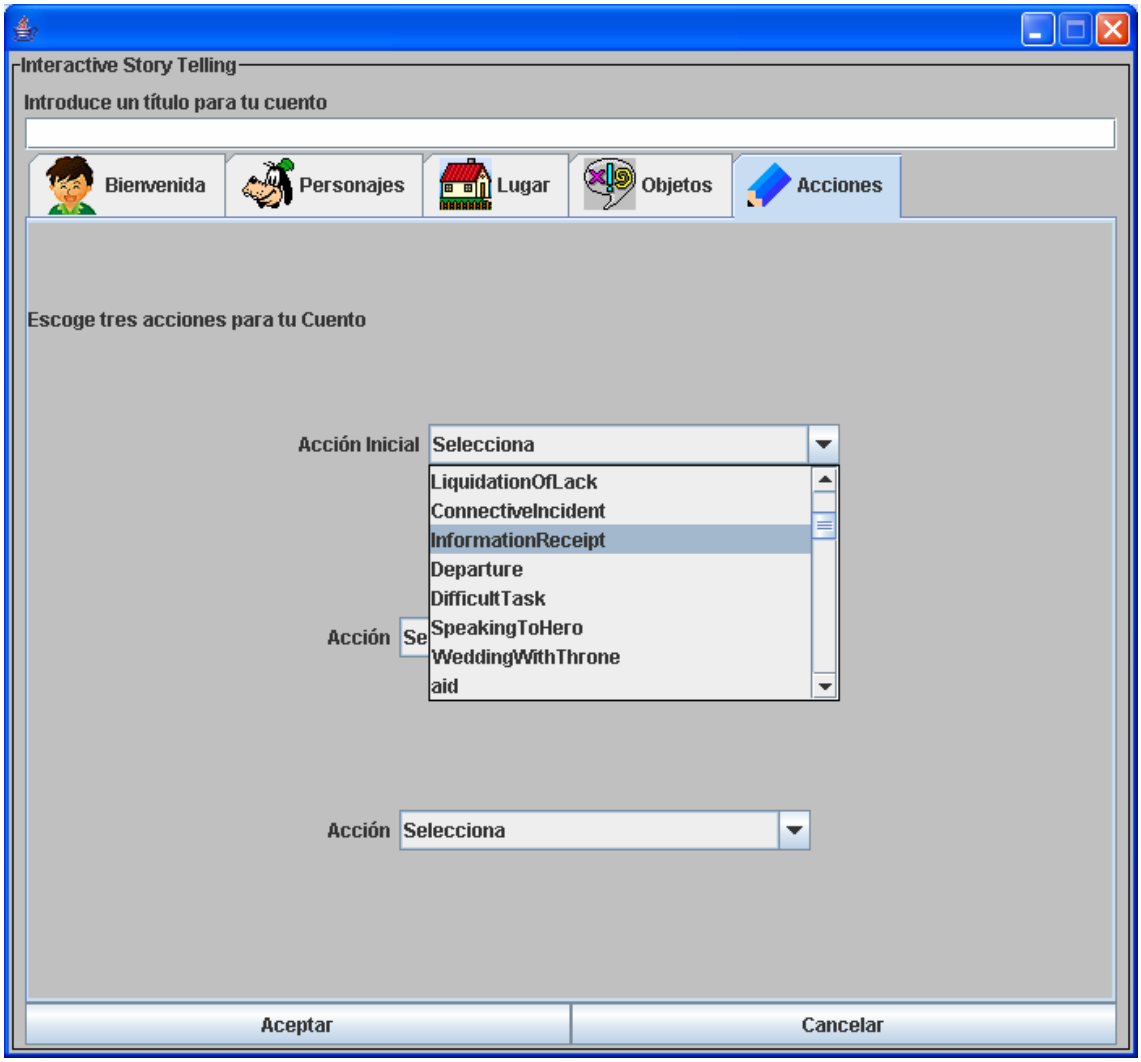


Figura 4.5: Pestaña de elección de acciones.

Deberemos poner un nombre a nuestro cuento en el campo de texto que aparece en la parte superior del formulario y ya están todos los elementos necesarios para poder construir el cuento.

Durante la generación de nuestro cuento, en determinadas ocasiones tendremos la oportunidad de elegir qué acción queremos que sea la siguiente en suceder. En este caso aparecerá un nuevo formulario que nos mostrará lo que ha sucedido en nuestro cuento hasta el momento y una pantalla de elección con las dos posibles acciones siguientes. Deberemos elegir cuál de esas acciones queremos que suceda.

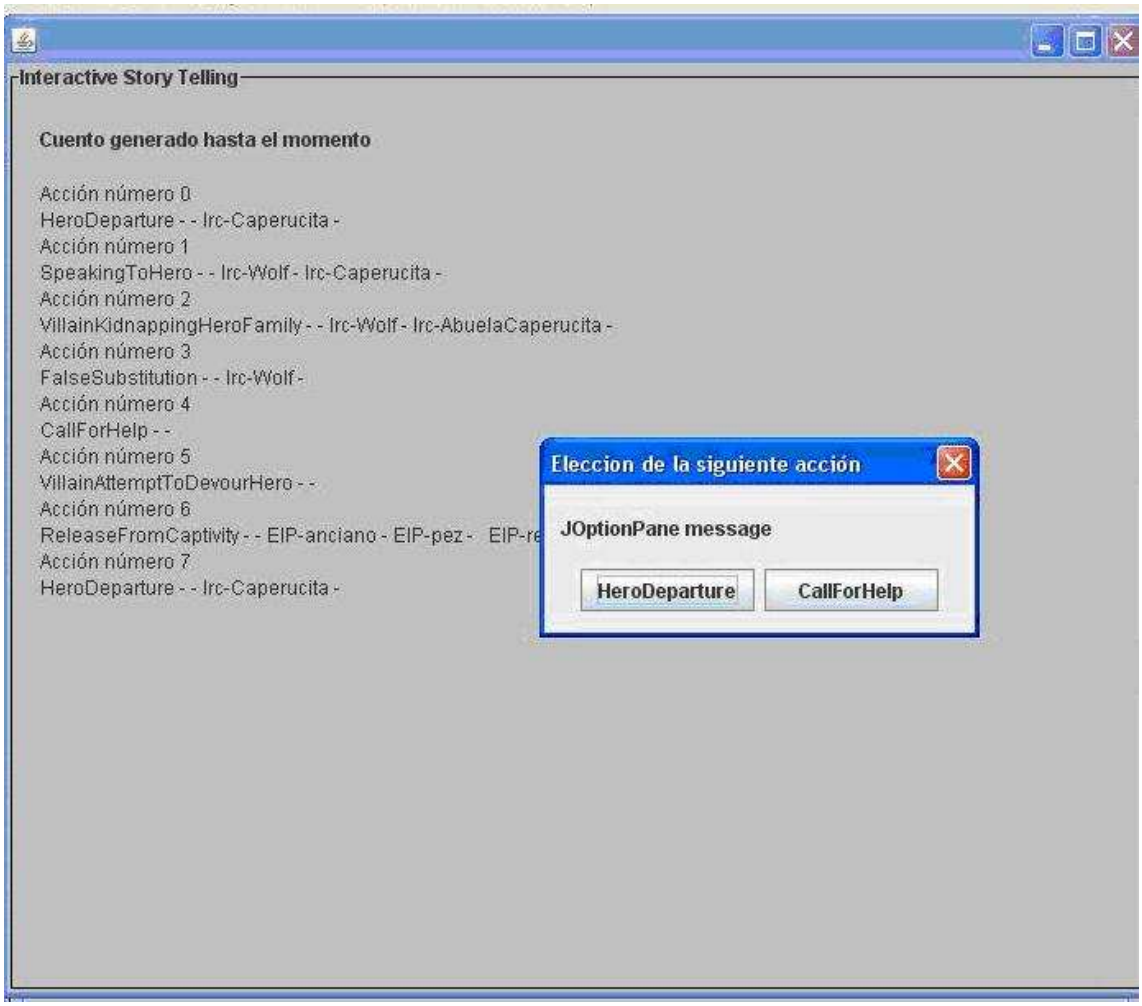


Figura 4.6: Interactividad del sistema

Finalmente, en la nueva pestaña aparecerá el cuento final con todos sus elementos.

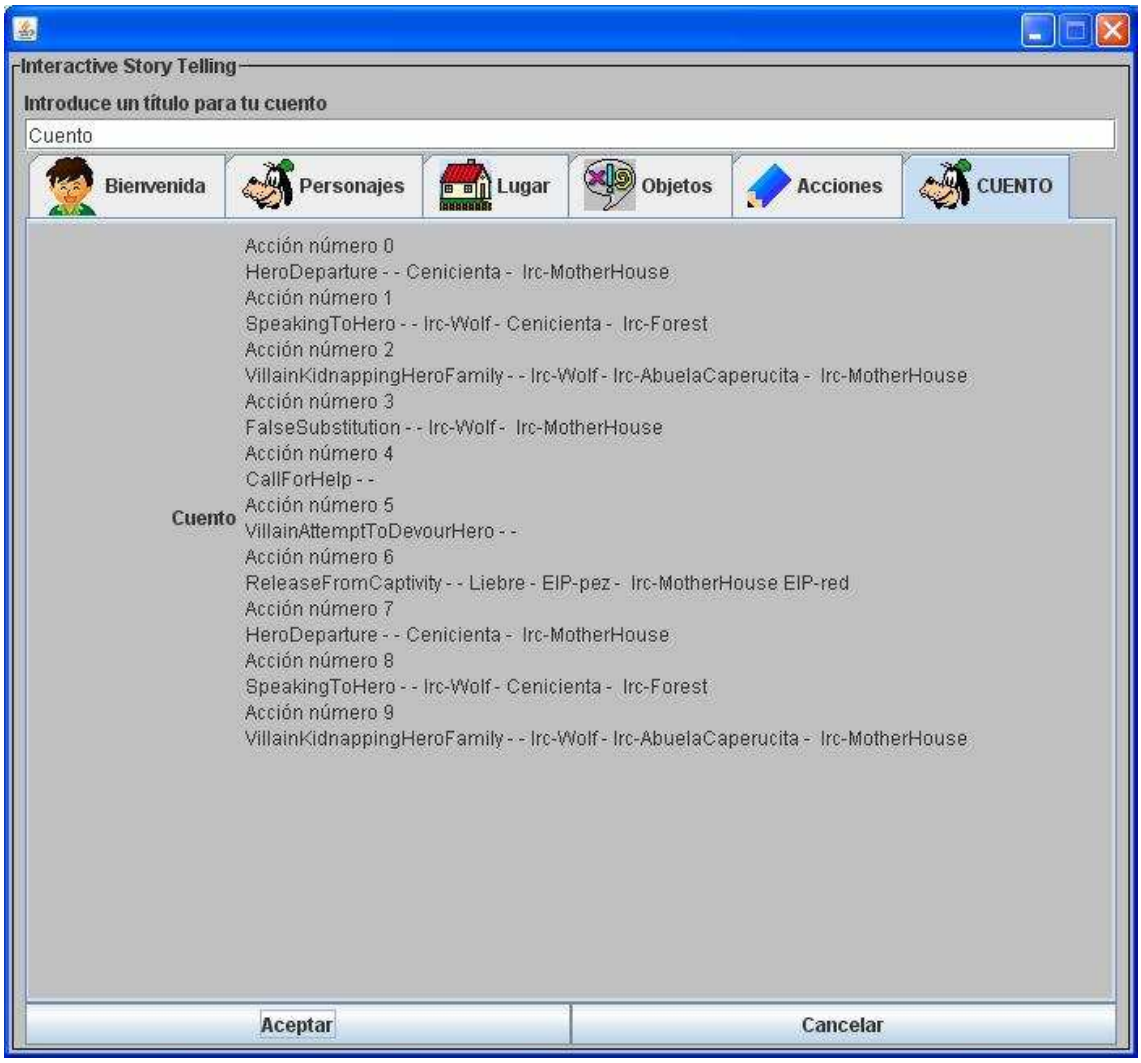


Figura 4.7: Resultado de generación del cuento.

4.2 Razonamiento CBR convencional

Nuestro proyecto tiene un sistema de razonamiento mediante el cual se van generando las diferentes historias. Éste se basa en la idea de los sistemas CBR explicadas en el capítulo 1.

El primer tipo de razonamiento implementado se basa en el ciclo CBR transformacional que ya fue explicado en el apartado 1.3.1. Recordamos que se basaba en una recuperación del caso de la base de casos que más se parecía al problema o consulta. En nuestro sistema, la consulta son los datos de entrada introducidos por el usuario, es decir, los personajes que quiere para su cuento, el lugar, el objeto y por último las acciones. Estos son los datos que componen la query y que serán usados por la función de similitud, explicada en el punto 4.4.

En nuestro caso, se van comparando uno a uno los cuentos de la base de casos con los datos de la query de usuario hasta quedarnos con el caso que más elementos en común tiene con el problema actual. Ese caso recuperado será adaptamos posteriormente para que incluya todos los elementos de la query de usuario (si no son todos al menos los máximos posibles ya que habrá ocasiones en las que no será posible que aparezcan todos ya que el cuento perdería su sentido). La manera de adaptar el cuento se detalla en el apartado 4.5.

Esta recuperación presenta dos claros inconvenientes. El primero de ellos, y que se explica extensamente en la discusión del capítulo 6, es que al realizar de este modo la recuperación, el cuento no es muy creativo ya que se recupera el cuento entero cada vez y lo único que cambiará serán las características que introdujo el usuario. El segundo inconveniente es la eficiencia. Cada vez que se desea recuperar un caso hay que comparar la query de usuario con todos los cuentos que se encuentran almacenados en la base de casos. Si la base de casos no está muy completa y los casos son pocos, esto no es un gran problema. Pero en general, lo interesante es que la base de casos esté muy poblada, y de ser así la eficiencia de la recuperación sería bastante mala.

4.3 Razonamiento basado en Adaptación Constructiva

En nuestro sistema no utilizaremos el ciclo CBR convencional, ya que consideramos que los resultados que podamos obtener serían muy similares a los ya incluidos en la base de conocimiento, no favoreciendo la creatividad en las historias[13].

El método empleado será un razonador basado en casos pero creado con "Adaptación Constructiva" [18]. Este método en contra del clásico es considerado como un método generativo y no transformativo, eso quiere decir que la solución al problema es construida en lugar de transformada.

A continuación vamos a explicar el proceso de razonamiento basado en Adaptación Constructiva de nuestro sistema.

[13] En el capítulo de Conclusiones podemos encontrar una comparativa de generación de un cuento utilizando el ciclo CBR convencional y utilizando Adaptación Constructiva.

Podría entenderse dicha técnica en términos abstractos, como una búsqueda en el espacio de soluciones donde los casos son usados en dos fases principales: generación de hipótesis y ordenación de hipótesis. Para llevar a cabo dicha búsqueda es necesario elegir uno de los algoritmos de búsqueda guiada que conocemos en la actualidad.

Nuestro sistema se basa en el funcionamiento de un algoritmo A* genérico dividido en las dos partes que se diferencian en la adaptación constructiva, generación de hipótesis y ordenación de hipótesis.

GENERACIÓN DE HIPÓTESIS

El algoritmo A* implementado se basa en la creación de nodos con la información de los casos, que van expandiéndose a lo largo del desarrollo del mismo. La información contenida en los nodos es la siguiente:

1. Una estructura de tipo cuento que en cada nodo, contendrá información acerca de las acciones, personajes, lugares y objetos rellenos en cada paso. En el caso inicial se crea el primer nodo con la estructura de tipo cuento construida a partir de la información recogida en la query.
2. Unas variables de control para indicar en qué paso de la construcción nos encontramos, y para determinar el final del proceso.

En nuestro caso necesitamos conseguir una historia o cuento y para ello debemos ir completando una serie de acciones, personajes, lugares y objetos que son los componentes en los que hemos dividido nuestras historias. Para nosotros serán los casos con los que trabaja el razonador de sistema.

Decidimos un orden para la realización del razonamiento. Puesto que las acciones del cuento implican una serie de personajes que las realizan o padecen, lugares en los que se desarrollan y objetos implicados decidimos que este fuera el orden de razonamiento. Por lo tanto se comenzará por las acciones y se continuará por los personajes, lugares y objetos.

Partimos, como hemos comentado anteriormente, de la idea de que un cuento es una estructura compuesta de las siguientes partes:

- Lista de acciones
- Lista de personajes
- Lista de lugares
- Lista de objetos

Por lo tanto, a lo largo del proceso de generación de hipótesis, debemos ir rellenando estas listas, tomando como datos de partida los de la query inicial obtenida a partir de las elecciones realizadas por el usuario al inicio del sistema de generación del cuento.

Hemos definido la query inicial como un cuento que contiene únicamente las elecciones iniciales del usuario que constituyen las restricciones impuestas al resultado del sistema. El usuario elige tres tipos de acciones que se corresponden con la primera acción del cuento que quiere obtener y otras dos acciones que deberán aparecer a lo

largo del desarrollo del mismo. Nuestro usuario elige también dos personajes del cuento, así como un tipo de lugar y un tipo de objeto que incluir en el cuento resultado. Es necesario matizar que las acciones, lugares y objetos están organizados en la ontología (base de conocimiento) jerárquicamente dependiendo de su tipo y que el usuario elije de qué tipo deben ser, nunca instancias concretas. Las instancias concretas se irán creando durante el proceso de razonamiento partiendo del tipo elegido o inferido en cada caso.

El razonamiento comienza rellenando las acciones y, por lo tanto, comenzará rellenando una acción del tipo indicado por el usuario como acción inicial. El resto de acciones se consiguen de tres maneras diferentes hasta llegar a un número máximo de acciones que compondrán una historia (fijado en diez).

- Puesto que, como ya hemos mencionado en el capítulo anterior, las acciones tienen la propiedad *postAccion* que nos indica la acción posterior a esa en el cuento representado en la base de conocimiento, normalmente nos quedamos con el tipo de la instancia que se encuentra en la propiedad mencionada de la última acción incluida en nuestro cuento a generar. A partir de este tipo podemos volver a aplicar el razonamiento para crear la nueva acción a incluir.

- Las acciones que el usuario ha decidido que deben aparecer a lo largo del cuento se deberán incluir detrás de alguna cuando la función de similitud entre la *postAccion* de la última incluida y la que queremos incluir nos lo permita.

- A lo largo de la recuperación nos encontramos con ocasiones en las que la elección que realiza el algoritmo de la acción que sigue a la última incluida encuentra un "empate" en el valor de la función de similitud utilizada como heurística. En estos casos se presentan al usuario las posibilidades para que elija una de ellas. De este modo solucionamos el problema de continuidad y cumplimos uno de los objetivos de este sistema que es la interactividad usuario-sistema durante todo el desarrollo del cuento.

Durante el relleno de las diez acciones que componen el cuento se deben incluir en las tres listas restantes aquellos personajes, lugares u objetos que intervienen en las acciones rellenadas. Este es el motivo de empezar con las acciones puesto que son las únicas que incluyen dependencias con respecto al resto. Cuando decimos incluir no nos referimos a introducir los personajes, lugares y objetos concretos sino, en el caso de los personajes incluimos que necesitamos personajes de unos roles determinados, en el caso de los lugares y los objetos incluimos que necesitamos instancias de lugares y objeto de un tipo determinado. Una vez rellenas las acciones el razonador deberá recorrer las listas de personajes, lugares y objetos para ir rellenando cada uno de los incluidos, es decir, a partir de los roles y tipos almacenados se deberán conseguir instancias concretas accediendo a la ontología y eligiendo una de las instancias de esas clases aplicando el algoritmo A*[14] basado en la similitud establecida de antemano.

[14] Algoritmo A*, algoritmo de búsqueda heurística.

ORDENACIÓN DE HIPÓTESIS

Una vez generadas las hipótesis en un paso del algoritmo, lo que significa que se han expandido las acciones, personajes, lugares u objetos posibles para elegir en la siguiente iteración del algoritmo, es necesario ordenar las mismas. Este proceso traducido a detalles de implementación consiste en la actualización de las dos listas que manipula el algoritmo A*.

La lista de nodos abiertos es aquella en la que se van introduciendo todas las posibilidades expandidas en cada iteración del algoritmo. Esta lista está ordenada de acuerdo a la heurística de los nodos de manera que siempre será primero en la lista el nodo de menor coste heurístico.

La lista de nodos cerrados, en la que se almacenan los nodos que han sido explorados, es decir, que se han utilizado para continuar la expansión a partir de ellos. De esta forma se evitarían ciclos en la ejecución del algoritmo.

ANÁLISIS DEL ALGORITMO EN NUESTRO SISTEMA

Según lo explicado en el capítulo 1, el algoritmo A* presenta unas características específicas que debemos analizar.

La característica más importante es que, como todo algoritmo de búsqueda guiada mediante una función heurística, es necesario que la misma sea admisible para el correcto funcionamiento del algoritmo. Para cumplir esta admisibilidad se debe cumplir que el coste estimado nunca sea mayor que el coste real de conseguir el objetivo desde el nodo evaluado en cada paso del algoritmo. En nuestro caso este hecho no es tenido en cuenta puesto que no buscamos la solución más prometedora en cada caso sino la que mejor se ajuste a los datos introducidos por el usuario en la query de entrada al sistema. En ningún momento nuestras heurísticas miden el coste de conseguir el objetivo sino la similitud del cuento que llevamos construido en cada iteración del algoritmo con el cuento que queremos construir. Por ello podríamos decir que el comportamiento del algoritmo en nuestro sistema se asemeja al de un A normal.

Una parte muy importante a tener en cuenta del algoritmo A* es el coste computacional y espacial del mismo. La complejidad computacional de un algoritmo depende de la heurística usada y su capacidad para llegar al objetivo en el menor tiempo posible. En nuestro caso el coste computacional no depende de la heurística en sí, sino de la base de conocimiento del sistema. A mayor cantidad de casos en la base de conocimiento, menor será la complejidad computacional del sistema puesto que hay más posibilidades de encontrar casos muy similares al buscado. Sin embargo, a medida que se aumentan los casos en la base de conocimiento se empeora la complejidad espacial porque se expanden más nodos para cada paso del algoritmo.

MODIFICACIONES EN EL ALGORITMO A*

Debido a la necesidad de introducir interactividad en nuestro sistema, debíamos encontrar la manera de detener el proceso de generación de hipótesis por parte del algoritmo para introducir puntos de decisión de los que tomara parte el usuario del sistema. Así tuvimos que modificar a nuestras necesidades el comportamiento general del algoritmo A* de la siguiente manera.

Primero, la forma de determinar que un nodo es objetivo. Un nodo será objetivo si ya tiene completas todas las listas de acciones, personajes, lugares y objetos con valores válidos. Hemos fijado que el número máximo de acciones que compondrán un cuento sean diez, así si ya hemos rellenado diez acciones pasamos a buscar otro elemento (siguiendo el orden establecido). Pero también marcaremos como que hemos terminado de rellanar acciones, si la acción actual es una de las que no tiene post-acción (aunque el número de acciones que hayamos rellenado sea menor de diez), ya que esto nos indica que es una función de resolución y es el final del cuento.

Segundo, la elección del siguiente nodo no siempre la realiza el algoritmo, en determinadas ocasiones, cuando la similitud entre dos nodos es menor que un epsilon determinado, entra en juego la interactividad del sistema. Es el momento de que el usuario elija qué es lo siguiente que quiere que pase en su cuento. Una vez que el usuario ha escogido la acción obligamos al algoritmo a que escoja el nodo correspondiente a la elección del usuario como siguiente nodo a expandir.

Y por último, otra alteración del algoritmo es, que cada vez que hemos completado una de las listas que hay que rellenar, ya sea de acciones, personajes o lugares, vaciamos de la lista de nodos abiertos todos los nodos correspondientes a la iteración anterior. Es decir, si hemos terminado de buscar acciones y pasamos a buscar personajes, de la lista de nodos abiertos eliminaremos todos los nodos correspondientes a acciones, para evitar posibles vueltas atrás del algoritmo.

CONSTRUCCIÓN DE UN NUEVO CASO EN LA BASE DE CONOCIMIENTO

Por último, después de haber generado el cuento, tenemos que este constituye un nuevo caso válido para introducir en nuestra base de conocimiento, de esta forma permitimos que el sistema vaya adquiriendo conocimiento a la vez que genera cuentos. La forma de crear un nuevo caso se hace de igual manera que en el *Editor de Cuento*s.

4.4 Heurística y función de similitud

En este apartado vamos a explicar las heurísticas y las funciones de similitud usadas para las dos recuperaciones.

Como hemos explicado anteriormente, la heurística de un sistema de razonamiento se puede definir como una serie de funciones cuyos valores son utilizados por el razonador para poder tomar las decisiones pertinentes a lo largo del proceso de razonamiento. Las funciones deben ser capaces de evaluar los conceptos requeridos en cada problema. En nuestro caso, la parte del sistema correspondiente a la heurística es fundamental en el razonamiento del mismo así como en la generación futura del cuento.

ADAPTACIÓN CONSTRUCTIVA

En este método el concepto de heurística es muy importante puesto que la construcción del cuento paso a paso depende únicamente de los resultados heurísticos para cada nodo. El método de razonamiento heurístico seguido corresponde a un sencillo razonamiento hacia atrás. A pesar de no ser un razonamiento hacia atrás exacto, podemos basarnos en esta idea por el motivo de que partimos de una serie de objetivos, que son los almacenados en la query de entrada del sistema, y que deseamos cumplir al máximo de las posibilidades. Luego, a partir del resultado que se quiere obtener se elige el camino a seguir para lograrlo. En base a esta idea se implementan las funciones heurísticas[15].

La heurística del sistema se basa en una serie de funciones de similitud que definiremos a partir de algunas propiedades de los conceptos e instancias representadas en la ontología. Este valor heurístico se obtiene realizando la suma del valor del coste de la solución hasta el momento y el coste supuesto hasta el objetivo, siguiendo la teoría del algoritmo A*. Para calcular el coste estimado desde el nodo actual hasta el nodo resultado final utilizamos unas medidas de similitud del nodo actual con el resultado que queremos obtener.

Las funciones de similitud del sistema se basan en comparar la solución que hemos conseguido en cada momento de cómputo con la deseada, teniendo en cuenta que la solución deseada está guiada, en cierto modo, por las elecciones del usuario. Así, hay una serie de comprobaciones que componen la valoración de la similitud y que otorgan mayor o menor igualdad entre nodos. Podemos dividir las funciones de similitud en dos grupos, un primer grupo dedicado a calcular la similitud entre acciones y otro grupo dedicado a valorar los nodos que resulten de añadir personajes, lugares y objetos. Como ya hemos explicado en otras ocasiones, las acciones que compondrán un cuento son la parte principal del mismo y el resto de conceptos dependen de las restricciones impuestas por dichas acciones, esto supone que dedicaremos más esfuerzo a calcular la similitud entre acciones.

[15] *"Si no encuentras la solución, haz como si ya la tuvieras y mira qué puedes deducir de ella (razonando hacia atrás)"* George Pólya.

Similitud entre acciones

La función de similitud de las acciones entra en funcionamiento cuando el algoritmo está rellenado acciones en cada nodo que expande y se basa en comprobar si la acción que se introduce en ese nodo tiene algún concepto en común con los elegidos por el usuario. Entendemos por conceptos los que definen nuestros cuentos; acciones, personajes, lugares y objetos. Por lo tanto el nodo irá sumando determinadas penalizaciones dependiendo de los siguientes casos:

1. La acción introducida no tiene como post-acción alguna de las elegidas por el usuario para formar parte del cuento

De este modo intentamos darle prioridad a aquellas acciones cuya post-acción pertenezca al mismo tipo de alguna de las que el usuario quiere incluir en el cuento resultado. De esta manera garantizamos que se cumplirá, en la medida de lo posible, la elección del usuario siempre y cuando el sentido del cuento generado se preserve. Si introducimos las acciones elegidas por el usuario de modo aleatorio el sentido del cuento puede perderse y decidimos que el sentido primara sobre la garantía de cumplir las ideas del usuario.

2. La acción introducida no implica ningún personaje cuyo rol corresponda con alguno de los elegidos por el usuario.

Así garantizamos que los roles elegidos por el usuario sean los primeros en aparecer en el cuento resultado y sólo se añadan más personajes en caso de ser necesarios. De esta manera, los roles elegidos por el usuario siempre aparecen en el cuento resultado a excepción de que el usuario elija dos personajes con el mismo rol, en cuyo caso aparecerá el primero de ellos.

3. La acción introducida no implica ningún lugar que corresponda con el tipo de alguno de los elegidos por el usuario.

Con esto tenemos que las acciones que se desarrollen en un lugar del tipo del elegido por el usuario pertenezcan al cuento desarrollado por el sistema, siempre y cuando esto sea posible. En la fase de relleno de lugares, su función heurística intentará escoger el lugar concreto.

4. La acción introducida no implica ningún objeto que corresponda con alguno de los elegidos por el usuario.

De este modo garantizamos que las acciones impliquen algún objeto cuyo tipo coincida con el tipo del objeto elegido por el usuario siempre y cuando las acciones que compongan el cuento necesiten algún objeto para llevarse a cabo.

Similitud entre personajes

La función de similitud de los personajes es usada por el algoritmo A* implementado cuando cada nuevo nodo generado expande un nuevo personaje. En el caso de los personajes las acciones rellenadas previamente restringen los roles que deben tener los personajes del cuento. La función de similitud comprobará si el personaje que se añade al nodo tiene un rol de los elegidos por el usuario, o por el contrario es un personaje cuyo rol es necesario en la historia. Se realiza de esta manera debido a que el usuario tiene dos posibilidades de elección en el caso de los personajes. Por un lado puede elegir un personaje existente en la base de conocimiento, en este caso la función de similitud juega un papel muy importante ya que es necesaria para garantizar la aparición del personaje en concreto y no sólo el rol, cosa que ya se tiene en cuenta en la elección de las acciones. La otra opción del usuario es elegir el rol que quiere que desempeñen sus nuevos personajes a los que nombrará según su gusto. En este caso, la función de similitud no es necesaria debido a que el relleno de las acciones ya garantiza la aparición de los roles de la query y la adaptación modifica los nombres según la elección del usuario.

Similitud entre lugares

En el caso de los lugares la función de similitud es muy sencilla. Durante el proceso de inclusión de las acciones se determina el tipo de lugar, de modo que en esta fase la función de similitud sólo determinará si el lugar es exactamente el elegido por el usuario.

Similitud entre objetos

En el caso de los objetos la función de similitud vuelve a ser sencilla. Durante el proceso de inclusión de las acciones se determina el tipo del objeto de modo que en esta fase, la función de similitud sólo determinará si el objeto es exactamente el elegido por el usuario.

CICLO CBR CONVENCIONAL

En este método el concepto de heurística no es tan importante puesto que la construcción del cuento no se realiza paso a paso, como en CA, sino que se hace una evaluación heurística al inicio que recupera un cuento entero para pasar a la posterior adaptación del mismo. El método de razonamiento heurístico seguido se corresponde con intentar abordar primero un problema general, recuperamos un cuento completo que se asemeje al que queremos conseguir, para después adaptarlo consecuentemente[16]. De esta manera sólo es necesaria una función de similitud y se utiliza al inicio del proceso de recuperación, no durante el proceso de manera repetida como en CA.

[16] *"Intenta abordar primero un método más general (es la "paradoja del inventor": el propósito más ambicioso es el que tiene más posibilidades de éxito)* George Pólya.

La función de similitud en este caso permite la recuperación del cuento que más se parece al que pretende conseguir el usuario. Para ello, compara las acciones que componen cada uno de los cuentos de la base de conocimiento con las introducidas por el usuario en la query inicial. El rol de los personajes que intervienen en cada cuento también se compara con el de los elegidos por el usuario así como el tipo de los lugares y los objetos. Una vez recuperado el cuento más similar se procede a su adaptación exacta con la query introducida por el usuario.

4.5 Adaptación del cuento construido

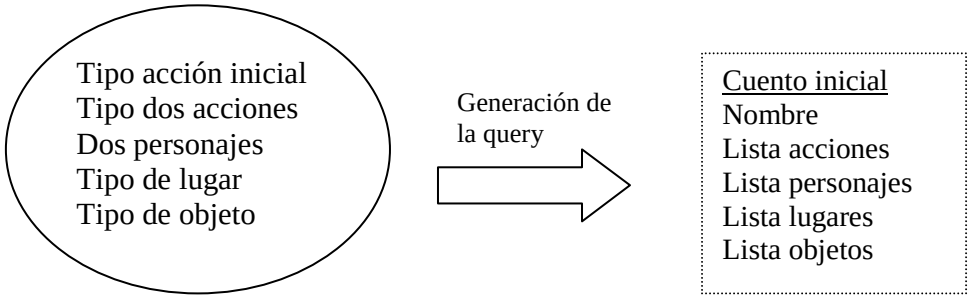
Una vez que el algoritmo A* ha terminado, debemos comprobar si la información que el usuario introdujo como datos iniciales en la query aparece en el cuento final. De no ser así habrá que modificar en el cuento ciertos elementos para incluir los elegidos por el usuario, sin que ello modifique el significado del cuento ni haga que se pierda el sentido.

La adaptación que llevamos a cabo comienza por cambiar el nombre de los personajes del cuento por los nombres que el usuario introdujo[17]. Al terminar la generación, buscamos el personaje que tiene el mismo rol que el personaje de la query inicial y cambiamos su nombre por el nuevo nombre elegido para él. Este proceso se repite con el segundo personaje de la query inicial. A continuación se cambia el nombre al resto de personajes implicados en el cuento. Después pasamos a adaptar los lugares del nuevo cuento. Esto no se va a hacer siempre ya que como la función de similitud guía la búsqueda, habrá ocasiones en las que directamente aparezca el lugar elegido en el cuento. De no ser así vamos a modificar alguno de los lugares del cuento resultante. Para ello debemos buscar en la ontología cuál es el concepto padre del tipo de lugar de la query[18]. En la ontología se busca cuál es la clase padre del tipo de lugar y se almacena cuáles son todos los hijos de esa clase padre, es decir, todos los lugares "hermanos" del lugar que queremos adaptar. Tras esto, buscamos en la lista de lugares del cuento final si su tipo es "hermano" del de la query. De serlo cambiamos el lugar que aparecía en el resultado por alguna de las instancias del tipo elegido por el usuario. Por ejemplo, si el usuario eligió como tipo de lugar *Castle* y este no apareciese en el resultado buscamos los lugares "hermanos" de *Castle* que son: *House* y *Palace*. Buscaríamos en la lista de lugares del cuento si alguno de ellos es de tipo *House* o *Palace*. De haberlo, sustituiríamos ese lugar por alguna de las instancias del tipo de lugar de la query, es decir, por alguna instancia de *Castle*.
Después hay que adaptar los objetos. Estos se dividen en inanimados y animados. Por lo tanto, lo primero que hay que hacer es ver de qué tipo es el objeto que eligió el usuario. Luego se busca en la lista de objetos que aparecen en el cuento algún otro lugar del mismo tipo y se sustituye por el de la query.

[17] Simplemente recordar que al principio cuando el usuario elige los datos iniciales para el sistema, también decide los nombres para los personajes que selecciona.
[18] Recordemos que el usuario no elige un lugar concreto para su cuento sino un tipo de lugar.

4.6 Esquema del proceso de razonamiento

A partir de las elecciones del usuario se crea la Query o cuento inicial:



Elecciones del usuario

En cada paso del algoritmo se van añadiendo acciones, personajes, lugares u objetos de la siguiente manera:

Acciones

1) Accedemos a la lista de acciones

```
ListaAcciones
```

2) Buscamos la última rellena

```
ListaAcciones:
     accion1
     accion2
     accion3
     accion4
     accion5←última acción rellena
     accion6
```

3) Accedemos a su "postaccion"

```
Accion6:
     TipoAccion
     Nombre
     PreAccion
     PostAccion← nos quedamos con esta postAccion
     Lugar
     Objeto
```

4) Buscamos el tipo de la postAccion

```
PostAccion:
     TipoAccion←nos quedamos con su tipo
     Nombre
     PreAccion
     PostAccion
     Lugar
     Objeto
```

5) Buscamos todas las instancias en la ontología de la clase correspondiente al tipo de la

```
postAccion:
     TipoAccion:
          Instancias:
               instancia1
               instancia2
               instancia3
               instancia4
```

6) Para cada una de las instancias creamos el nodo correspondiente y lo insertamos en la lista que recibirá el proceso de selección del algoritmo. De esta manera terminamos el proceso de adquisición de posibilidades.

Nodo1 nodo2 nodo3 nodo4

Cada uno de los nodos tiene la siguiente información relevante dentro de la variable cuento que contienen:

```
Nodo1:
     ListaAcciones:
          accion1
          accion2
          accion3
          accion4←instancia1
Nodo2:
     ListaAcciones:
          accion1
          accion2
          accion3
          accion4←instancia2

Nodo3:
     ListaAcciones:
          accion1
          accion2
          accion3
          accion4←instancia3
```

```
Nodo4:
     ListaAcciones:
          accion1
          accion2
          accion3
          accion4←instancia4
```

En este momento se introducen los nodos en la lista que recibe la parte de selección de hipótesis. En este paso se calcula la heurística para cada uno de los nodos de la lista de abiertos y se elige el más prometedor. A partir del nodo más prometedor se repite el proceso descrito accediendo a su lista de acciones rellenas.

Personajes

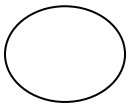
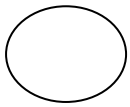
1) Accedemos a las lista de personajes:

```
ListaPersonajes:
      personaje1
            Tipo: tipo1
            Nombre: null
      personaje2
            Tipo: tipo2
            Nombre: null
      personaje3
            Tipo: tipo3
            Nombre: null
      personaje4
            Tipo: tipo4
            Nombre: null
```

2) Accedemos al primer personaje no relleno

```
      ListaPersonajes:
            personaje1←primer personaje no relleno
```

3) Buscamos en la ontología todos los personajes de este tipo y creamos un nodo nuevo con ese personaje.

Nodo1 Nodo2

La información de cada uno de los nodos será igual a excepción de la lista de personajes:

```
Nodo1:
ListaPersonajes:
            personaje1:
                  Tipo: instancia1
            Nombre: null

Nodo2:
      ListaPersonajes:
            personaje1:
                  Tipo: instancia2
                 Nombre: null
```

Una vez que ya tenemos toda la información de los casos pasamos a la parte de selección de hipótesis.

Lugares

1) Accedemos a las lista de lugares:

```
ListaLugares:
lugar1
      Tipo: tipo1
      Nombre: null
lugar2
      Tipo: tipo2
      Nombre: null
```

2) Accedemos al primer lugar no relleno

```
ListaLugares:
      lugar1←primer lugar no relleno
```

3) Buscamos en la ontología todos los lugares de este tipo y creamos un nodo nuevo con ese lugar concreto.

Nodo1 Nodo2

La información de cada uno de los nodos será igual a excepción de la lista de lugares:

```
Nodo1:
ListaLugares:
           lugar1:instancia1

Nodo2:
      ListaLugares:
           Lugar2:instancia2
```

Una vez que ya tenemos toda la información de los casos pasamos a la parte de selección de hipótesis.

Objetos

1) Accedemos a las lista de objetos:

```
ListaObjetos:
objeto1
      Tipo: tipo1
      Nombre: null
objeto2
      Tipo: tipo2
      Nombre: null
```

2) Accedemos al primer objeto no relleno

```
ListaObjetos:
      objeto1←primer objeto no relleno
```

3) Buscamos en la ontología todos los objetos de este tipo y creamos un nodo nuevo con ese objeto concreto.

Nodo1 Nodo2

La información de cada uno de los nodos será igual a excepción de la lista de objetos:

```
Nodo1:
ListaObjetos:
          objeto1:instancia1

Nodo2:
      ListaObjetos:
          objeto2:instancia2
```

Una vez que ya tenemos toda la información de los casos pasamos a la parte de selección de hipótesis.

En la parte de selección de hipótesis se calcula la heurística para cada nuevo nodo dependiendo de si se trata de un relleno de acción, personaje, lugar u objeto. Se introducen los nodos en la lista de nodos abiertos y se extrae el primero (menor coste heurístico). Podemos apreciar ahora por qué es necesario vaciar esta lista cuando se pasa de un tipo de relleno a otro. Imaginar una situación en la que estamos rellenando personajes pero la lista de nodos abiertos no ha sido vaciada al pasar de rellenar acciones a personajes:

```
Lista de nodos abiertos
      Nodo1: relleno acción
      Nodo2: relleno personaje
      Nodo3: relleno personaje
```

Como vemos se escoge un nodo de relleno de acción de manera que se volverá a rellenar la última acción que ya ha sido rellena anteriormente con un nodo más prometedor. Este hecho es debido a que no se ha vaciado la lista y se han quedado nodos de relleno de acciones que no fueron elegidos en su iteración por no ser los más prometedores de su iteración pero que tiene menor heurística que los nodos correspondientes a la iteración actual.

La parte de adaptación, tan sólo adapta los valores obtenidos a los de la query, por ejemplo el nombre de los personajes dependiendo del rol.

5. GENERACIÓN DE LENGUAJE NATURAL

La última parte de este trabajo se corresponde con la forma de generar texto a partir de la estructura obtenida para el guión del cuento y de esta forma poder mostrárselo al usuario.

Los cuentos generados por nuestro sistema están representados en la estructura denominada *TransferTale*, que se describió en la sección 3.3.1 y podemos observar en la figura 3.24, pero esta estructura no es adecuada para mostrar al usuario el resultado de sus elecciones. Por lo tanto es necesario narrar el cuento obtenido.

La narrativa es una parte muy compleja al implicar una serie de propiedades lingüísticas difíciles de reproducir computacionalmente. Cabe mencionar que esta parte no formaba parte de los objetivos del proyecto, ya que ella misma podría constituir un proyecto entero hemos generado una forma sencilla de narración para poder mostrar los resultados de forma más clara al público.

La narración de nuestro sistema se ha desarrollado mediante la técnica del texto enlatado. De esta manera cada uno de los tipos de acciones está traducido a lenguaje natural en un fichero de texto guardado con el nombre del tipo de acción al que corresponda. Para llevar a cabo la narración sólo es necesario acceder al fichero de la acción aplicada en cada caso y concatenar su descripción con los personajes, lugares y objetos que forme parte de la misma. De esta forma podemos obtener un texto para nuestros cuentos aunque lingüísticamente hablando no sea muy correcto, ya que el conocimiento del sistema no está orientado a este efecto.

6. EVALUACIÓN

6.1 Adaptación Constructiva *vs* Ciclo CBR convencional

Un objetivo fundamental del proyecto era conseguir cuentos originales y con sentido. Por este motivo realizamos dos implementaciones distintas de razonamiento del sistema, la adaptación constructiva y el ciclo CBR convencional. A continuación se analizan los dos procedimientos con sus ventajas y desventajas.

6.1.1 Resultados Adaptación Constructiva

Al generar un cuento con adaptación constructiva, el cuento se va construyendo paso a paso desde la acción inicial que elige el usuario. El paso siguiente se genera gracias al tipo de la post-acción. Esto hace que cualquier instancia de ese tipo sea una posible candidata a siguiente acción del cuento, independientemente de que en la ontología dichas acciones pertenezcan a cuentos completamente distintos. De la misma manera que se pueden mezclar acciones de diferentes historias también pueden aparecer personajes, lugares u objetos de cuentos diferentes. A la hora de elegir las acciones, personajes, lugares y objetos del cuento se tienen en cuenta los datos introducidos por el usuario en la query de entrada. De esta manera la creatividad es mayor ya que cada vez que se cambie un elemento de la query el resultado final puede cambiar dando lugar a un cuento distinto.

Query

Personajes→ cenicienta, cenicientaStepMother
Lugar→ forest
Objeto→ pan
Acción inicial→ heroDeparture
Acciones→ wedingOrThrone,

```
Erase una vez, en un país
lejano vivían Cenicienta,
CenicientaStepMother, AbuelaCaperucita y
FairyGodMother.Un buen día Cenicienta parte
a un lugar lejano desde un lugar
llamado MotherHouse. CenicientaStepMother
habla con Cenicienta en un lugar llamado
Forest. CenicientaStepMother encierra a
AbuelaCaperucita en un lugar llamado
MotherHouse. CenicientaStepMother se hace
pasar por otra persona en un lugar llamado
MotherHouse. El protagonista pide ayuda. El
protagonista consigue liberarse del
cautiverio. FairyGodMother decide dar un
escarmiento a CenicientaStepMother en un
lugar llamado Forest.
```

Figura 6.1: Resultado en lenguaje natural del cuento.

En la figura 6.1 comprobamos como, a partir de la primera acción de la query de entrada que almacena las elecciones del usuario, se genera el cuento correspondiente. Las elecciones del usuario se intentan cumplir en la medida de lo posible. En el ejemplo comprobamos como los personajes que aparecen en el cuento incluyen los elegidos por el usuario, así ocurre con uno de los lugares en los que se desarrollan las acciones que componen el cuento. Sin embargo no se incluye el objeto porque las acciones que componen el cuento no necesitan ningún objeto para su ejecución y tampoco se pueden incluir las dos acciones que ha elegido el usuario porque esto ocasionaría una pérdida de coherencia de la producción. Las heurísticas del sistema son las responsables de que las elecciones del usuario puedan o no cumplirse para cada producción.

Query

Personajes→Estrella en el rol de cenicienta, Carmen en el rol de cenicientaStepMother
Lugar→ House
Objeto→ Shoe
Acción inicial→ Kidnapping
Acciones→Punishment, FalseSubstitution

Erase una vez, en un país lejano vivían anciano, FairyGodMother, Carmen, Estrella y AbuelaCaperucita.Un buen día anciano es secuestrado por FairyGodMother en un lugar llamado Forest con un Inanimate. Carmen habla con Estrella en un lugar llamado Forest. Carmen encierra a AbuelaCaperucita en un lugar llamado MotherHouse. Carmen se hace pasar por otra persona en un lugar llamado MotherHouse. El protagonista pide ayuda. El protagonista consigue liberarse del cautiverio. anciano decide dar un escarmiento a Carmen en un lugar llamado MotherHouse.

Figura 6.2: Resultado en lenguaje natural del cuento.

En la figura 6.2 comprobamos cómo se cumplen las elecciones del usuario al completo. Este hecho es debido a que las funciones heurísticas del sistema han determinado que el cumplimiento de los datos del usuario no interfiere en el sentido del resultado final.

En la adaptación constructiva hemos incorporado la interactividad logrando todavía mayor grado de creatividad ya que el usuario elige en ciertos momentos cuál es el tipo de acción que quiere que suceda a continuación. En función de las elecciones que haga el usuario el cuento podrá ser muy diferente el resultado final del cuento. Incluso aunque la query inicial elegida por el usuario fuese completamente igual en distintas ejecuciones, las decisiones que más tarde tendrá que tomar el usuario durante el desarrollo de la generación harán que cambien algunos de los personajes, lugares y objetos del cuento, pero sobre todo las acciones, consiguiendo cuentos distintos.

Sin embargo, la adaptación constructiva tiene el inconveniente de que, al construirse el cuento paso a paso, en ciertos momentos, bien por el nodo elegido por el algoritmo o por la decisión del usuario, es posible que el sentido del cuento se pierda un poco. Por eso es tan importante la función de similitud que nos proporciona heurísticas diferentes a los nodos, de manera que el algoritmo elija aquellos nodos que mejor se ajusten en cada momento al desarrollo de la historia teniendo en cuenta todos los elementos. Otro de los inconvenientes que aparecen es que si la base de cuentos no es muy amplia y los diferentes tipos de acciones de la ontología no tienen muchas instancias puede suceder que los cuentos que resulten sean muy parecidos a los de la ontología. Este problema va desapareciendo cuanto más poblada esté la ontología.

Las siguientes figuras muestran un ejemplo de las diferentes producciones que puede obtener el sistema a partir de la elección del usuario cuando se produce un cuento interactivo. La query de entrada formulada por el usuario es la siguiente:

Query

Personajes→cenicienta,fairyGodMother
Lugar→ Forest
Objeto→ Shoe
Acción inicial→ LiquidactionOfLack
Acciones→VillainKidnappingHeroFamily, DifficultTaskWithSolution

Figura 6.3: Query de entrada el sistema.

El sistema sólo coloca la primera acción del cuento que va a producir, esta primera acción corresponde con la que ha introducido el usuario del sistema en la query como acción inicial. En la siguiente iteración se debe producir un empate entre los nodos que se pueden expandir y por ello el sistema muestra al usuario un panel ilustrativo del estado de la generación realizada hasta el momento y las opciones a desempatar, figura 6.4. La elección del usuario va a determinar la continuación del cuento y, por lo tanto, el resultado final. Si elegimos la primera opción el resultado se muestra en la figura 6.5. Como podemos observar se produce una nueva interactividad, eligiendo la primera opcion obtenemos el resultado de la figura 6.6

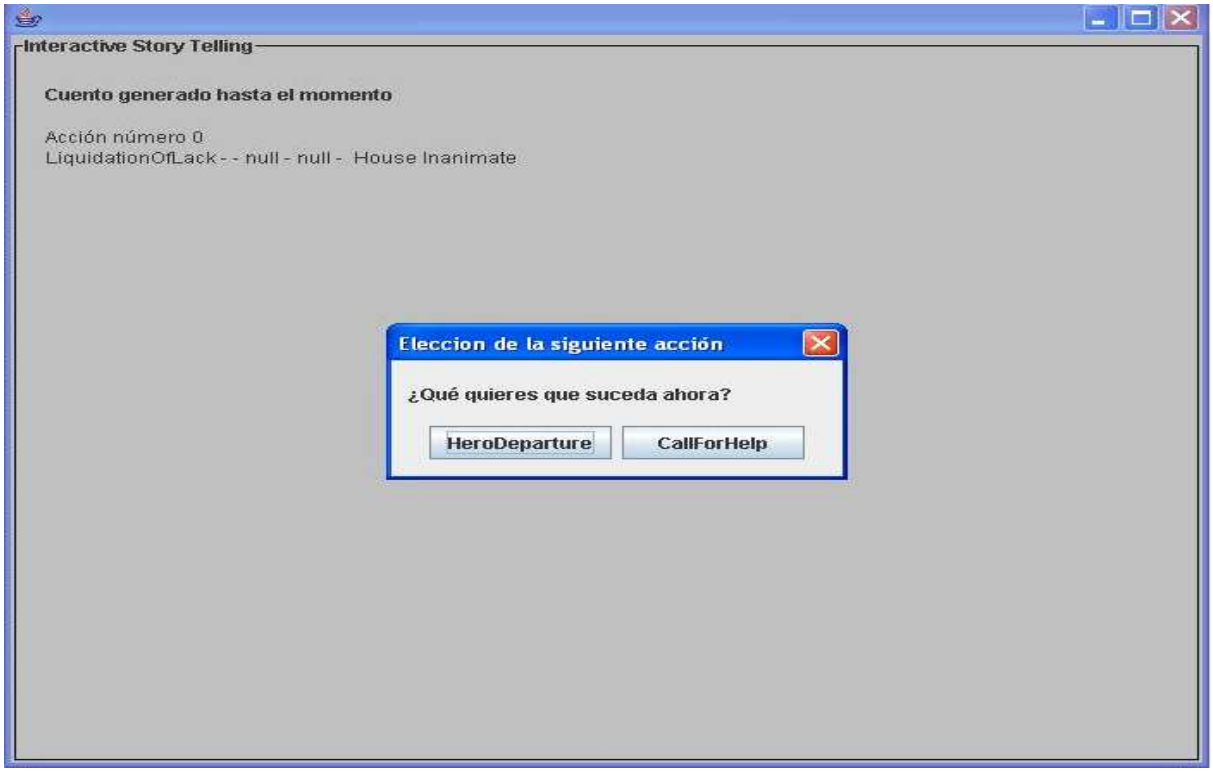


Figura 6.4: Muestra del panel de interactividad I.

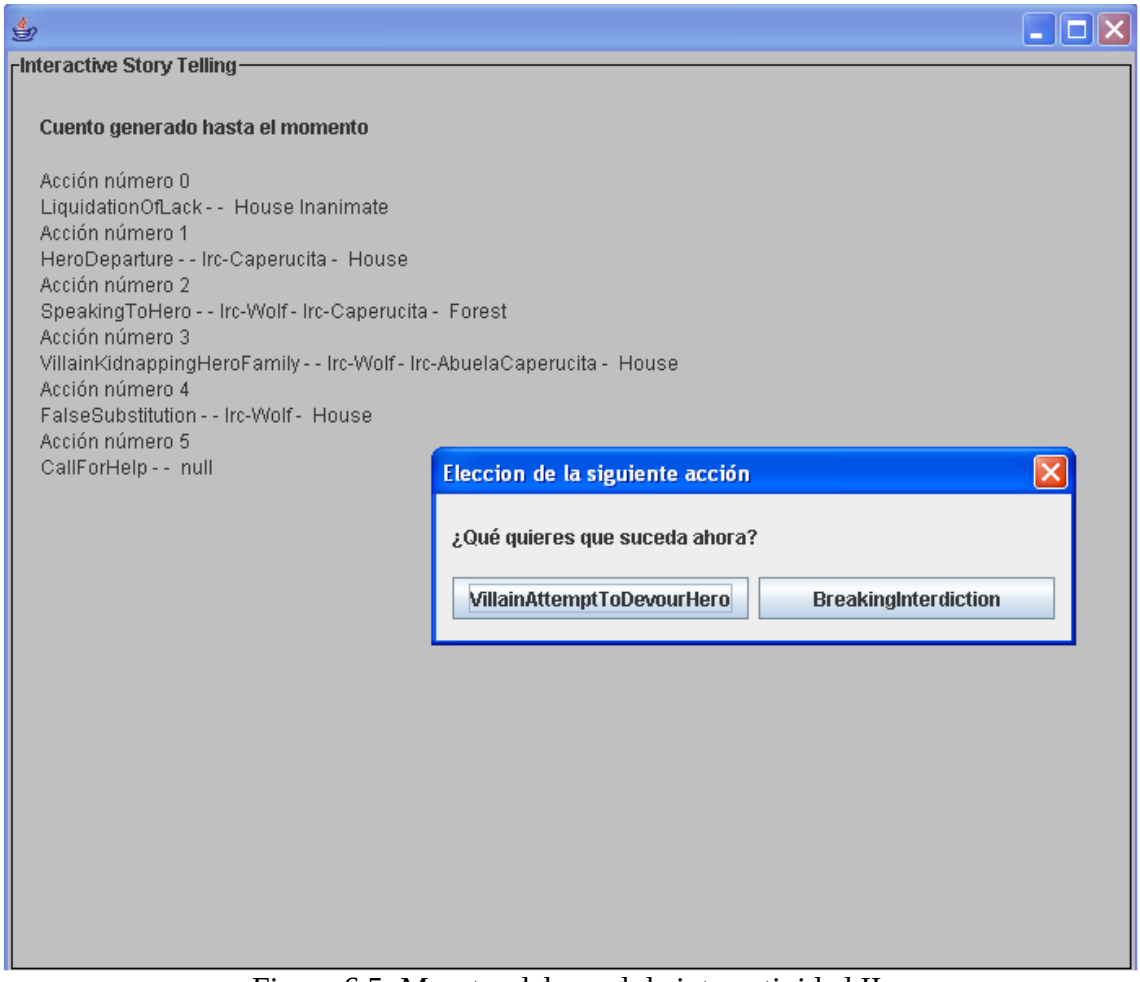


Figura 6.5: Muestra del panel de interactividad II.

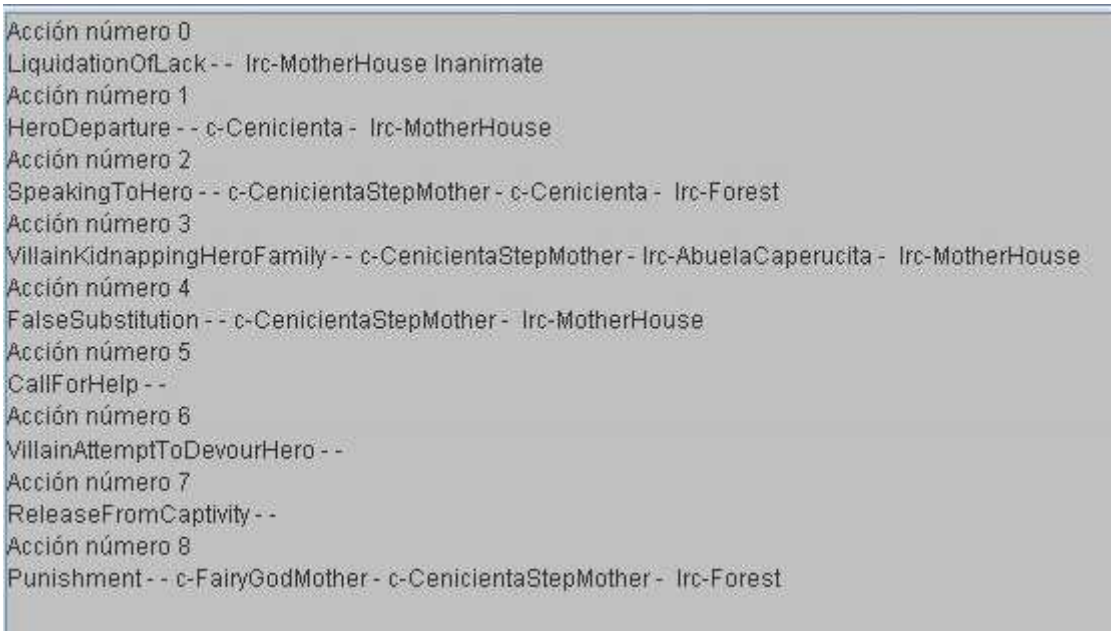


Figura 6.6: Resultado I.

Sin embargo, si en la última iteración mostrada, la correspondiente a la figura 6.5, elegimos la segunda opción, el resultado obtenido es el mostrado en la figura 6.7.

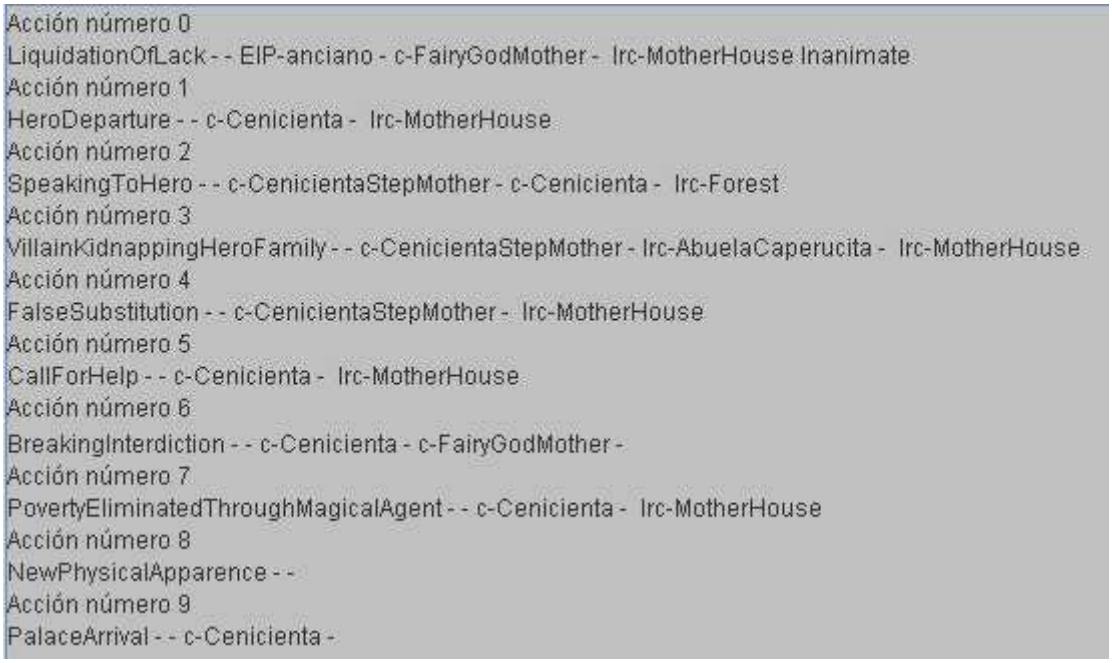


Figura 6.7: Resultado II.

Otra opción sería elegir la segunda de las propuestas en la primera interactividad que se produce, la correspondiente a la figura 6.43. En ese caso el resultado es el mostrado a continuación.

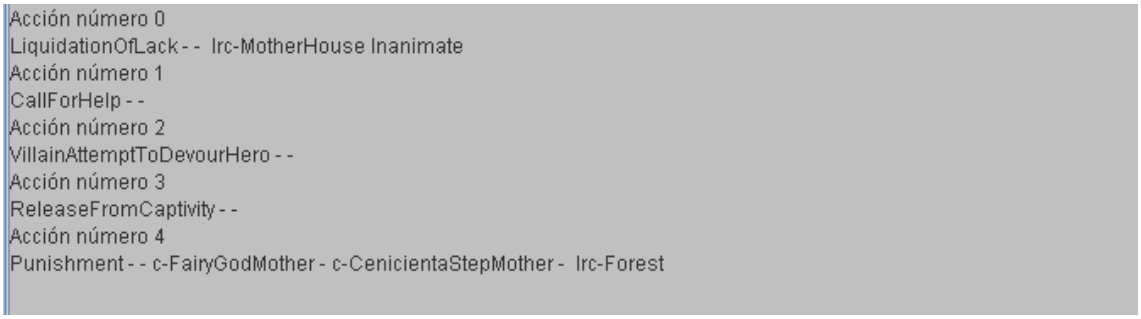


Figura 6.8: Resultado III.

6.1.2 Resultados Recuperación mediante ciclo CBR convencional

Esta recuperación consiste en comparar los elementos elegidos por el usuario en la query con los cuentos representados en la ontología. Vamos comparando uno a uno hasta quedarnos con el cuento que más elementos en común tiene con la query. Recuperamos ese cuento entero de la ontología y pasamos a adaptar la información de la query que no aparezca en el cuento como se explicó antes.

Los resultados obtenidos mediante la recuperación con el ciclo CBR convencional son mucho menos expresivos y creativos que los resultados obtenidos mediante la adaptación constructiva ya que lo que hacemos es recuperar el cuento completo y aunque luego lo adaptemos con las definiciones de la query, esto no es suficiente como para que el cuento y su significado cambien notablemente. Los cuentos finales obtenidos de esta manera son prácticamente iguales a los que hay en la ontología. Además, en este caso, el usuario no decide en ningún momento cuál es la siguiente acción que va a suceder por lo tanto las historias que aparecen no tienen prácticamente ninguna modificación.

Como también sucede con la adaptación constructiva cuantos menos cuentos haya en la ontología las posibilidades de recuperación por parte del sistema disminuyen, es decir, por normal general recuperará siempre los mismos cuentos ya que no habrá mucha variedad donde elegir. Sin embargo, con la recuperación la coherencia de los cuentos no se pierde en ningún momento.

Vamos a analizar a continuación los ejemplos mostrados en el apartado de la adaptación constructiva pero aplicando el proceso del ciclo CBR sobre ellos.

Query

Personajes→ Estrella en el rol cenicienta, Carmen en el rol cenicientaStepMother
Lugar→ forest
Objeto→ pan
Acción inicial→ heroDeparture
Acciones→ wedingOrThrone,

Erase una vez, en un país lejano vivían Estrella, Carmen, caperucita, lobo, AbuelaCaperucita y leñadores.Un buen día Estrella parte a un lugar lejano desde un lugar llamado MotherHouse. Lobo habla con caperucita en un lugar llamado Forest. Lobo encierra a AbuelaCaperucita en un lugar llamado GrannyHouse. Lobo se hace pasar por otra persona en un lugar llamado GrannyHouse. El protagonista pide ayuda. El protagonista consigue liberarse del cautiverio. Leñadores decide dar un escarmiento a Carmen en un lugar llamado Forest.

Figura 6.9: Resultado en lenguaje natural del cuento.

Podemos observar mediante el ejemplo de la figura 6.9 que el cuento recuperado corresponde con el de caperucita que se encuentra representado en la base de casos del sistema. Tan sólo se adapta el nombre de los personajes en función del rol que realizan dentro del mismo. No hay riqueza en la producción al no combinar partes de cuentos diferentes de la query produciendo una falta absoluta de creatividad.

Query

Personajes→ Estrella en el rol cenicienta, Carmen en el rol cenicientaStepMother
Lugar→ House
Objeto→ Shoe
Acción inicial→ Kidnapping
Acciones→ punishment, falseSubstitution

Erase una vez, en un país lejano vivían Estrella, Carmen, caperucita, lobo, AbuelaCaperucita y leñadores.Un buen día Estrella parte a un lugar lejano desde un lugar llamado MotherHouse. Lobo habla con caperucita en un lugar llamado Forest. Lobo encierra a AbuelaCaperucita en un lugar llamado GrannyHouse. Lobo se hace pasar por otra persona en un lugar llamado GrannyHouse. El protagonista pide ayuda. El protagonista consigue liberarse del cautiverio. Leñadores decide dar un escarmiento a Carmen en un lugar llamado Forest.

Figura 6.10: Resultado en lenguaje natural del cuento.

Como podemos observar en el ejemplo propuesto en la figura 6.10, al igual que en el ejemplo de la figura 6.9, el resultado es menos rico en personajes que el obtenido mediante el método de la adaptación constructiva al no mezclar acciones, objetos, lugares o personajes de otros cuentos. Además en este caso no se comienza por la acción elegida por el usuario.

6.1.3 Conclusión

Podemos observar que el proceso de generación de un cuento mediante Adaptación Constructiva es mucho más creativo frente a la utilización del ciclo CBR convencional, ya que las posibilidades de elección del siguiente paso no están limitadas a la similitud con el caso inicial, sino que dependen del anterior, al ir eligiendo en función del tipo de la post-acción, y porque las posibilidades de combinación de las acciones de un cuento con los demás componentes de otro cuento distinto son mayores con CA que con el ciclo CBR convencional.
Los resultados obtenidos mediante CA son mejores ya que la mezcla de los ingredientes que componen un cuento se realiza en base a todos los cuentos que se encuentran en la base de casos, obteniendo así un cuento más rico en variedad de personajes, acciones, lugares y objetos. Esta riqueza también es debida a que el método de CA es un método generativo y no transformacional –como ya explicamos en la sección 4.3-, es decir la solución se va construyendo en función de lo anterior, y la recuperación se realiza paso a paso, en cada momento recuperamos el caso más parecido, al contrario que en el transformativo que primero recupera el caso más parecido y después lo transforma.

6.2 Ejemplos de ejecución del sistema

A continuación mostraremos algún ejemplo de ejecución del sistema. Y veremos cómo a partir de unos datos iniciales el sistema es capaz de generar una nueva historia.

Ejemplo 1

Como datos de partida del problema tenemos:

- Personaje 1*: lrc-Caperucita* a la que hemos llamado Niña
- Personaje 2: *LA-Ayudantes* a los que hemos llamado Amigos
- Lugar: *Forest* (bosque)
- Objeto: *shoe* (zapato)
- Acción inicial: *HeroDeparture* (partida del héroe)
- Acción 1: *InformationReceipt* (recibir información)
- Acción 2: *VillainKidnappingHeroFamily* (el malo secuestra al a familia del héroe)

Tras la generación de cinco acciones aparece la interactividad. Debemos elegir por dónde queremos que continúe nuestro cuento.

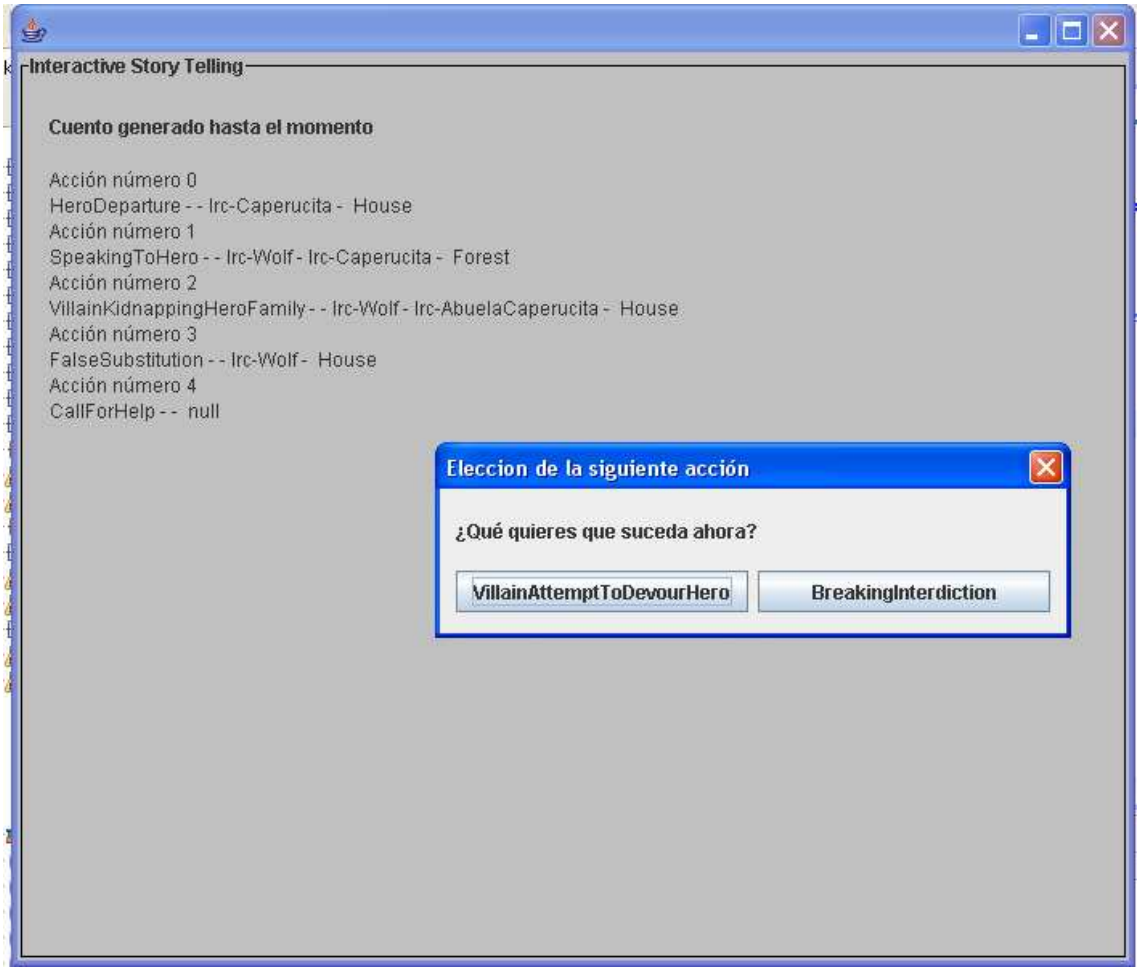


Figura 6.11: Interactividad del ejemplo 1

Tenemos dos opciones, *VillainAttemptToDevourHero (*el malo quiere comerse a nuestro héroe) o *BreakingInterdiction* (o rompemos una prohibición).
Seleccionamos la primera opción, queremos que el malo intente comerse a nuestro héroe. El resultado final del cuento es el siguiente.

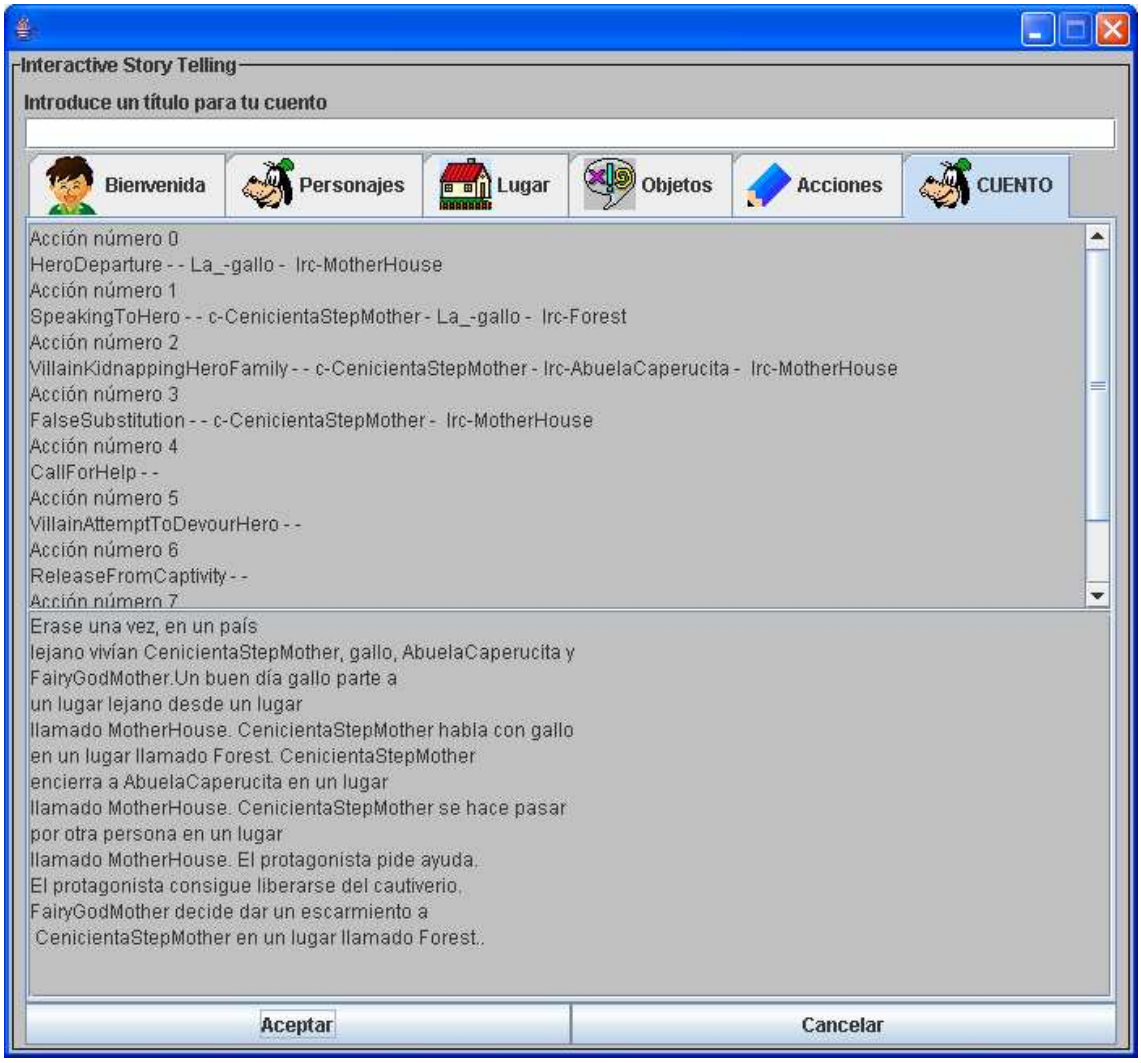


Figura 6.12: Resultado final del ejemplo 1

Obtenemos un cuento en el que nuestro protagonista (*niña)* parte de su casa. En el bosque se para a hablar con alguien que se encuentra, resultando ser este un villano. Después este villano encierra en la casa del héroe a un familiar, y se hace pasar por otra persona para despistar al héroe. Nuestro héroe viendo la situación decide llamar a sus *amigos* para que le ayuden a salvar a su familia. Se encuentra con un problema, el malo pretende comérselo, pero sus *amigos* llegan antes de que se lo coman y liberan al héroe (*niña)* y a su familia del cautiverio y entre todos le dan un escarmiento al malo.

Ejemplo 2

Como datos de partida del problema tenemos:

- Personaje 1*: lrc-Cenicienta* a la que hemos llamado Lucía
- Personaje 2: *LP-Pez* al que hemos llamado Miguel
- Lugar: *Castle* (Castillo)
- Objeto: *Horse* (Caballo)
- Acción inicial: *Arrival (*Llegada a algún lugar)
- Acción 1: *NewPhisicalApparence* (Alguien toma una nueva apariencia física)
- Acción 2: *WeddingOrThrone (*Hay una boda con posible ascenso al trono)

Este ejemplo es más sencillo que el anterior. En este caso, no aparece interactividad ya que de las acciones que se ven involucradas en la generación del cuento no hay más de una instancia. Tenemos que el cuento se genera automáticamente sin aparición de interactividad. Además en este caso tiene menos acciones porque al llegar a la acción de tipo *WeddingOrThrone*, la instancia de esta acción no tiene post-acción lo que provoca que el cuento termine ahí.

Figura 6.13: Resultado final del ejemplo 2

Obtenemos un cuento en el que Lucía, nuestra protagonista llega a un castillo y se ve involucrada en una persecución. Sale victoriosa y regresa a casa de su madre, a pesar de su nueva apariencia física obtenida en el castillo todo el mundo la reconoce. Y el cuento termina con que nuestra protagonista se casa con su perseguidor.

7. CONCLUSIONES Y TRABAJO FUTURO

7.1 Aportaciones del sistema

La aportación principal de este trabajo es la propuesta un sistema que fuera capaz de generar cuentos a través de un sistema de razonamiento, que a la vez fuera capaz de ser interactivo con el usuario, y que empleara una base de conocimiento sobre el dominio. Las aportaciones específicas son:

- En cuento a la base de conocimiento. Hemos conseguido una estructura sólida y estable y con posibilidades de ampliación de manera fácil. El dominio de esta base está restringido al conocimiento sobre el dominio en el que estamos trabajando, es decir, al dominio de los cuentos, limitado a los posibles elementos que pueden aparecer en un cuento. Creemos que con respecto a este dominio la base de conocimiento es rica en conceptos, así como en las propiedades necesarias para establecer relaciones con cada uno de los elementos constitutivos de una historia.

- Algo bastante interesante es que la base de conocimiento sea capaz de adquirir nuevos casos enriqueciendo el conocimiento del dominio. El editor de cuentos permite a través de una interfaz fácil y manejable, incluir nuevos casos sin necesidad de tener conocimientos sobre el sistema. El incluir un editor de cuentos ha facilitado nuestro problema inicial, era difícil obtener casos sobre los que poder razonar, casos que poblaran nuestra base de conocimiento, pero este editor facilitó ese trabajo, gracias a que personas ajenas al proyecto pudieron poblar la ontología.

- Con respecto al generador, la principal aportación es el sistema de razonamiento basado en Adaptación Constructiva y en el que hemos introducido la interactividad con el usuario, permitiendo así que éste se sienta partícipe en la creación del cuento.

7.2 Limitaciones del sistema y trabajo futuro

A continuación detallamos algunas limitaciones del sistema o campos donde se podría mejorar ciertas características, siendo estas mejoras y limitaciones propuestas de trabajo futuro.

- Con respecto a la *Base de conocimiento*:

  - Como hemos comentado la base de conocimiento tiene una estructura sólida y estable, pero el conocimiento del sistema está limitado al conocimiento del dominio. Pero a la vez este conocimiento para determinados casos es limitado, sobretodo en lo que se refiere a personajes y objetos de los cuentos. Actualmente sólo distinguimos que un personaje sea un *animal*, un *ser humano* o un *grupo de personas*, pero sería muy interesante y ayudaría a la creatividad de los cuentos que esta distinción se pudiera refinar más, por ejemplo distinguir entre niños, adultos y ancianos. También se podrían añadir cuestiones relativas a su estado social (rico o plebeyo). Otra cuestión seria poder relacionar los personajes entre sí, actualmente la ontología está preparada para soportar relaciones entre personajes, pero no lo usamos en ninguna representación.

  - Otro de los conceptos que hemos definido poco son los objetos, actualmente un objeto puede ser animado o inanimado. Los objetos pueden dar mucho juego en la creación narrativa y sería bueno que se pudiera refinar esta distinción en los objetos.

  - Por otro lado, también sería interesante poder incluir conocimientos lingüísticos para que la posterior generación del texto fuera más precisa. Actualmente sólo distinguimos entre el género de los personajes.

- Con respecto al *Editor de Cuentos*.

  - Algo que habría que revisar sería la eficiencia del programa, ya que para cuentos con un número considerable de elementos (acciones, personajes, lugares y objetos) el sistema tarda un tiempo en generar el fichero OWL.

  - Por otro lado, sería interesante que la creación de un personaje pudiera albergar más conocimiento, por ejemplo: edad del personaje, aspecto físico, etc… para que esto fuera posible, como hemos dicho antes habría que enriquecer el conocimiento.

  - En la interfaz de creación de una acción sería bueno poder ordenar el conjunto de funciones de Propp de alguna forma (orden alfabético).

- Otra posibilidad muy interesante seria que se añadiera la posibilidad de añadir comentarios a casi todo lo que creas. Hay muchas decisiones de diseño que se toman al vuelo y podría resultar interesante guardarlas en la ontología como comentario y para la propia consulta en el editor, para acordarnos de lo que hemos ido haciendo.

- Con respecto al *Generador de cuentos*:

  - Actualmente la generación del cuento se guía en base a la primera acción que elija el usuario y continúa con la post-acción que contiene esta primera acción. La forma de elegir la siguiente acción será aquella que tenga mejor función heurística de las diferentes posibilidades. Y estas posibilidades provienen de instancias válidas de la ontología pertenecientes a la función de Propp de la post-acción de la primera acción. Pero actualmente no tenemos en cuenta propiedades de las acciones tales como la causa y el efecto de una acción o la dependencia de una acción con respecto a otra. También se podría extender esta similitud entre dos acciones a funciones distintas de Propp.

  - Con respecto a la interfaz podría mejorarse el formulario que se muestra en el momento de elegir la siguiente acción del cuento, es posible que la forma de mostrar el estado actual del cuento no sea muy intuitiva.

  - Creemos que los objetos pueden añadir mucha creatividad a los objetos, y la adapatación de estos así como la utilización en las distintas acciones ha sido dejada un poco de lado.

  - Existe la posibilidad de que el sistema dé a elegir acciones que nos parezcan incoherentes con el desarrollo de la historia, por ejemplo puede darse el caso de tener que elegir que la siguiente acción sea la ruptura de una prohibición pero en nuestro cuento podemos observar que no tenemos ninguna prohibición, esto se debe a que las propiedades de la base de conocimiento relativas a la causa y al efecto de las acciones no se utilizan demasiado, un buen camino para generar mejores historias sería explotar las posibilidades de utilización de estas propiedades.

- Con respecto a la *generación del lenguaje natural*:

  - Este es un campo en el que no hemos profundizado demasiado, ya que dicha parte en sí misma podría constituir un único proyecto y porque no formaba parte de los objetivos iniciales.

8. BIBLIOGRAFÍA

[1] V. Propp. *Morphology of the Folktale*. University of Texas Press, 1968.

[2] R. Michael Young, *Notes on the Use of Plan Structures in the Creation of Interactive Plot*

[3] Andrew S. Gordon and Nicholas V. Iuppa, *Experience Management Using Storyline Adaptation Strategies*

[4] Belén Díaz Agudo. (Extracto de la tesis, Capítulos 2 y 5) *Cómputo de similitudes entre dos individuos almacenados en una ontología.*

[5] Matthew Horridge, Holger Knublauch, Alan Rector, Robert Stevens, Chris Wroe *A Practical Guide To Building OWL Ontologies Using The Protégé-OWL Plugin and CO-ODE Tools Edition 1.0*

[6] Federico Peinado and Pablo Gervás, *Evaluation of Automatic Generation of Basic Stories*

[7] Gervás, P. and Díaz-Agudo, B. and Peinado, F. and Hervás, R., *"Story Plot Generation based on CBR*

[8] Turner, S. R, *Minstrel: A Computer Model of Creativity and Storytelling*, Technical report UCLA-AI-92-04, Computer Science Department, University of California, USA, 1992.

[9] Federico Peinado and Pablo Gervás, *Minstrel Reloaded: From the Magic of Lisp to the Formal Semantics of OWL.*

[10] Marc Cavazza, Fred Charles, and Steven J. Mead, *Character-Based Interactive Storytelling* University of Teesside, UK

[11] Fairclough, C. and Cunningham, P., "A Multiplayer Case Based Story Engine", in *Proceedings of the 4th International Conference on Intelligent Games and Simulation*, EUROSIS, pp. 41-46, 2003.

[12] Malec, S. A., *Proppian Structural Analysis and XML Modeling*, http://clover.slavic.pitt.edu/~sam/propp/theory/propp.html

[13] Seifert, L., Lim, C., Tan, L. and Wee, N., *Digital Propp: Proppian Fairy Tale Generator*, Brown University, http://www.brown.edu/Courses/FR0133/Fairytale Generator/

[14] Peinado F., *Razonamiento Basado en Ontologías y Componentes Reutilizables para la Dirección Automática de Narración Digital e Interactiva*, Proyecto de Tesis Doctoral.

[15] Klein, S., Aeschlimann, J., Balsiger, D. F., Converse, S. L., Court, C., Foster,M., Lao, R., Oakely, J. D., and Smith, J. D. (1973). *Automatic novel writing.* Technical Report UWCS Tech Report No. 186, Deparment of Computer Sciences, University of Wisconsin Madison. An abridged version also appears in, Text Processing/Textverarbeitung. Edited by W. Burghardt & K. Hlker, pp.338–412, Berlin & New York: Walter de Gruyter, 1979.

[16] Lang, R. Raymond; 1997, *A formal model for simple narratives*; Ph.D. thesis; Tulane University. ftp://juno.eecs.tulane.edu/pub/lang/

[17] S. Bechhofer, F.van Harmelen, J. Hendler, I. Horrocks, D. L. Mc-Guinness, P. F. Patel-Schneider, and A. Stein. *OWL web ontology language reference, 2004.* http://www.w3.org/TR/2004/REC-owl-ref-20040210/.

[18] Enric Plaza and Josep-Lluís Arcos, *Constructive Adaptation*,ECCBR 2002:306-320

[19] Protégé Ontology Editor and Knowledge Acquisition System http://protege.stanford.edu/

[20] Sun-Microsystems. *Java products & technologies*, 2004. http://www.java.sun.com/

[21] *Ontobridge* http://gaia.fdi.ucm.es/grupo/projects/ontobridge/

[22] *Eclipse - an open development platform* http://www.eclipse.org/

[23] Oliver Zeigermann, Henrik Heine, *Java Search Library* http://jsl.sourceforge.net/

[24] *Resource Description Framework* (*RDF*) http://www.w3.org/RDF/

[25] http://www.ciudadseva.com/textos/cuentos/rus/afanasi/ana.htm

[26] Guha, R. V. & D. B. Lenat (1990). *"CyC: a Midterm Report*", Artificial Intelligence, 11: 32-59

[27] Hyopil Shin, *Mikrokosmos Ontology* http://crl.nmsu.edu/Research/Projects/mikro/htmls/ontology-htmls/onto.index.html

[28] Spibey, P. 1991, *"Express Language reference manual"* Technical Report ISO 10300 / TC184/SC4/WG 5, CADDETC.

[29] Schreiber, Th., Wielinga, B., Akkermans, J., Van de Velde, W. & de Hoog, R., 1994. *"CommonKADS: A comprehensive methodology for KBS development".*

[30] Gruber, T. R. (1993). *"A Translation Approach to Portable Ontologies*". Knowledge Acquisition, 5(2): 199-220.

[31] Schank, R., 1982. *Dynamic Memory*. Cambridge University Press.

[32] Bill Swartout, Ramesh Patil, Kevin Knight and Tom Russ. *"Toward Distributed Use of Large-Scale Ontologies"*. Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, November 9-14, 1996. Banff, Alberta, Canada.

[33] A. Bernaras, I. Laresgoiti, J. Corera, *Building and reusing ontologies for electrical network applications*, in: Proc. European Conference on Artificial Intelligence (ECAI'96), Budapest, Hungary, 1996, pp. 298–302.

[34] Kolodner, J. 1993. *Case-based reasoning*. Morgan Kaufmann Publishers, San Mateo California. 668 pp.

9. GLOSARIO

Cuento o historia: narración breve de ficción formada por un conjunto de acciones, personajes, lugares y objetos.

Generación Interactiva de Historias: Forma de generar una historia de manera que el usuario tome decisiones sobre las acciones posibles, en algunos puntos de su desarrollo.

Ontología: Base del conocimiento reutilizable, es decir, representación explícita del conocimiento, expresada en un lenguaje formal, acerca de un dominio.

Razonamiento Basado en Casos (CBR): Tecnología para la construcción de sistemas expertos alternativa a los sistemas basados en reglas. Los sistemas de razonamiento basado en casos son capaces de utilizar conocimiento específico de experiencias previas para resolver un nuevo problema. Capturan las características de dicho problema, buscan casos históricos con valores similares para dichas características, analizan las soluciones de estos casos y proponen una solución al problema, para, finalmente, aprender del problema actual para problemas futuros. Esto es lo que se conoce como ciclo CBR: Recuperar- Reutilizar- Revisar- Recordar.

Query: estructura de datos que almacena la información necesaria para producir un paso del razonamiento. La query inicial, con la que da comienzo el proceso de razonamiento se obtiene a partir de los datos introducidos por el usuario.

Adaptación constructiva: técnica basada en búsqueda para la reutilización generativa en los sistemas CBR para tareas de configuración. Este método en contra del clásico es considerado como un método generativo y no transformativo, eso quiere decir que la solución al problema es construida en lugar de transformada.

Acción: evento que sucede en un cuento. Dicho evento compromete a unos personajes que son los que se ven involucrados en el mismo, un lugar donde se desarrolla y en algunos casos un objeto necesario. Nuestros cuentos estarán compuestos por acciones. Estas acciones pueden identificarse con las funciones de Propp.

Personaje: ser que toma parte en la acción de un cuento. Puede ser animal, ser humano, agente o un grupo de agentes. A su vez estos personajes tendrán asociado un papel que desempeñarán en el transcurso de la historia.

Lugar: sitio que forma parte de un cuento.

Objeto: elemento necesario en algunos cuentos para llevar a cabo determinadas acciones. Estos objetos pueden ser animados (i.e.: un caballo) o inanimados (i.e.: una espada)

Lenguaje natural: en la filosofía del lenguaje, lenguaje hablado y/o escrito por humanos para propósitos generales de comunicación. El término lenguaje natural se refiere al estudio de las propiedades computacionales y de otro tipo implicadas en la comprensión, producción y uso de las lenguas naturales.

Algoritmo A*: algoritmo (A Estrella) de búsqueda guiada, se clasifica dentro de los algoritmos de búsqueda en grafos. El algoritmo encuentra, siempre y cuando se cumplan unas determinadas condiciones, el camino de menor coste entre un nodo origen y uno objetivo. El algoritmo utiliza una función de evaluación $f(n) = g(n) + h(n)$, donde $h(n)$ representa el valor heurístico del nodo a evaluar, y $g(n)$, el coste real del camino recorrido para llegar a dicho nodo.

Heurística: Según la Real Academia Española de la Lengua se entiende por heurística a la manera de buscar la solución de un problema mediante métodos no rigurosos, como por tanteo, reglas empíricas, etc.

Nodo: estructura de datos capaz de representar la información revelante para cada uno de los estados de un problema de búsqueda.

Nodo inicial: nodo de partida de una búsqueda.

Nodo objetivo: nodo final con éxito de una búsqueda.

Interactividad: diálogo entre el usuario y el sistema.