UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA
Departamento de Ingeniería del Software e Inteligencia Artificial

UN ARMAZÓN PARA EL DESARROLLO DE APLICACIONES DE NARRACIÓN AUTOMÁTICA BASADO EN COMPONENTES ONTOLÓGICOS REUTILIZABLES

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Federico Peinado Gil

Bajo la dirección del doctor
Pablo Gervás Gómez-Navarro

Madrid, 2008

- **ISBN: 978-84-692-0075-9**

Un armazón para el desarrollo de aplicaciones de narración automática basado en componentes ontológicos reutilizables

Tesis doctoral

Presentada por
D. Federico Peinado Gil

Dirigida por el
Prof. Dr. D. Pablo Gervás Gómez-Navarro

Facultad de Informática
Universidad Complutense de Madrid
Madrid, 2008

Un armazón para el desarrollo de aplicaciones de narración automática basado en componentes ontológicos reutilizables

Tesis doctoral

Presentada por
D. Federico Peinado Gil

Dirigida por el
Prof. Dr. D. Pablo Gervás Gómez-Navarro

Facultad de Informática
Universidad Complutense de Madrid
Madrid, 2008

Agradecimientos

"Si yo pudiera enumerar cuánto debo a mis antecesores y contemporáneos, no me quedaría gran cosa en propiedad". Estas palabras de Goethe resultan muy apropiadas para poner punto y final a una tesis. A lo largo de estos años he recibido aportaciones de mucha gente y aunque sólo mi nombre figura en la portada, no creo que exista una sola obra científica debida exclusivamente al mérito de su autor.

La primera persona en mi extensa lista de acreedores es Pablo Gervás, mi director de tesis. Su apoyo y colaboración durante este revuelto periodo de mi vida, en especial su ilusión durante las largas temporadas sin resultados tangibles, merecen todo mi agradecimiento.

Los sufridos coautores de nuestras publicaciones (Belén Díaz, Raquel Hervás, Virginia Francisco, Álvaro Navarro, Michael Santorum, Héctor Gómez, Pedro Pablo Gómez, Marco Antonio Gómez, Pablo Moreno, Birte Lönneker-Rodman, Jan Christoph Meister, Miguel Ancochea y Abraham López) han trabajado duro y a contrarreloj, por lo que merecen mi amistad y gratitud. Doy las gracias también a la pequeña comunidad de usuarios de tecnologías semánticas en que me he visto inmerso. Óscar Muñoz, Jose Miguel Mohedano, Marina Gallego, Irene Pérez y Almudena Ruiz han aportado mucho más de lo que piensan a construir esta tesis. También Francisco José Álvarez y Benedicto Rodriguez se merecen mi gratitud por orientarme en el mundo de las metodologías de desarrollo de ontologías, así como los compañeros de la excelente *European Summer School on Ontological Engineering and the Semantic Web* de 2006. El trabajo de los becarios y alumnos colaboradores del que, de una forma u otra, se beneficia esta tesis también ha de ser reconocido, como el de Carlos León, Jorge Carrillo-Albornoz o Susana Bautista. Gracias a todos por preguntar, corregir, informar y hacerme trabajar.

Recuerdo con emoción la buena acogida en el grupo de Marc Cavazza durante mis estancias de investigación en la Universidad de Teesside y en la Universidad de Hamburgo, con el grupo de narratología de Jan Christoph Meister. Reconozco verme inspirado a menudo por la dedicación y el esfuerzo de aquellos investigadores con los que he tenido la suerte de trabajar, como Fred Charles o Birte Lönneker-Rodman; también por la de aquellos que forman parte de lo que siento como mi comunidad científica (congresistas habituales de TIDSE, ICVS, etc.). He tenido la suerte de contar con colaboradores internacionales siempre dispuestos a todo, como Francisco Pereira y Amílcar Cardoso, dos verdaderos amigos.

Aunque son muchos los compañeros y profesores que han influido en mi trayectoria, quiero agradecer expresamente a Carmen Fernández y a Luis Hernández el abrirme la puerta al mundo académico, que ahora tan importante

es para mi, iluminando el algo oscuro sendero de la burocracia universitaria.

De una manera más formal, aunque no menos importante, quiero agradecer el apoyo moral y económico que me han ofrecido las siguientes instituciones: Grupos de investigación NIL, GAIA y el general ISIA de la Universidad Complutense de Madrid; Departamento de Ingeniería del Software e Inteligencia Artificial (antiguo departamento de Sistemas Informáticos y Programación) y Facultad de Informática de dicha universidad; Beca para Grupos de Investigación de la Universidad Complutense de Madrid y la Dirección General de Universidades e Investigación de la Comunidad de Madrid (UCM-CAM-910494) y Beca Predoctoral para la Formación de Personal Investigador de la Universidad Complutense de Madrid; Proyectos "Galante: Generación de lenguaje natural para textos con emociones" y "Ardisia: Arquitectura para el desarrollo e integración de simulaciones interactivas en el aprendizaje" del Ministerio de Educación y Ciencia (TIN2006-14433-C02-01 y TIN2005-09382-C02-01); proyecto "Javy Forest: Application of Metaphor and Blending in game environments" del Ministerio de Ciencia y Tecnología, en Acción Integrada Hispano-Portuguesa (HP2003-0068) entre la Universidad Complutense de Madrid y la Universidad de Coimbra, y proyecto "Arcano: Documentación de Armazones Basada en Casos y su Aprendizaje mediante Ejemplos" de la Comisión Interministerial de Ciencia y Tecnología (TIC2002-01961).

En la esfera de mi vida privada, hay alguien más a quien dar las gracias. Ella es Mª Ángeles, compañera inseparable que ha recorrido conmigo el tortuoso sendero de la carrera académica sin dudar de mi capacidad en ningún momento, a pesar de mis continuos descalabros.

El problema de las listas exhaustivas es que, en realidad, pocas lo son. La lista anterior no es ninguna excepción y por eso quisiera abarcar, con un amplio agradecimiento, a todos los compañeros, profesores, miembros del personal universitario, miembros del tribunal, investigadores que he conocido física o electrónicamente, congresistas, revisores, familiares y amigos cuyos nombres no puedo reunir aquí pero que sin duda también han sido "coautores espirituales" de esta tesis. Este último agradecimiento te incluye a ti, amigo lector, por estar dispuesto a dar sentido a mi trabajo.

Declaración

Todo el texto, los ejemplos, cuadros y figuras que aparecen en este trabajo y no aparecen acompañados de una referencia explícita a otra publicación, son creación original del autor.

Resumen

El ordenador se ha utilizado como medio de expresión narrativa desde hace décadas. La informática ha permitido desarrollar tecnologías muy potentes para el tratamiento de la información narrativa, permitiendo almacenar, organizar, modificar y reproducir contenidos de todo tipo. La automatización de algunas de estas tareas es perfectamente posible, aunque hay otras cuya formalización en términos computacionales se considera extraordinariamente compleja o incluso imposible, debido principalmente a que los procesos mentales que el ser humano emplea para acometerlas aún son grandes desconocidos. La generación de historias es una de esas tareas y su automatización forma parte de ese gran proyecto a largo plazo al que llamamos "inteligencia artificial".

Entre los años setenta y ochenta se llevaron a cabo numerosos estudios relacionados con la comprensión y la generación automática de historias que sentaron las bases para la investigación científica en esta materia. Tras años de desinterés por la cuestión, nuevas aplicaciones, enfoques, y condiciones técnicas favorables han provocado un ligero renacer del interés por las aplicaciones de narración automática en el ámbito científico y empresarial.

En este trabajo se estudian los fundamentos teóricos de la narración automática, se revisan las diversas metodologías y tecnologías implicadas en su desarrollo y se analizan los resultados obtenidos hasta el momento y documentados en la literatura científica.

Actualmente hay aplicaciones capaces de generar automáticamente el contenido, la estructura e incluso la presentación final (en forma de texto, gráficos, videos, etc.) de una historia. Sin embargo estas aplicaciones no revelan una solución única y sistemática para todos los problemas fundamentales de la generación de historias. Aunque hay excepciones, muchos de estos trabajos fallan en su planteamiento científico u obtienen soluciones poco generalizables, escalables, evaluables y útiles.

Este trabajo de tesis pretende construir un armazón sólido y adecuado para el desarrollo de aplicaciones de narración automática. Los objetivos se centran sobre aquellos problemas que afectan más negativamente a la organización, modularidad y reusabilidad de dos elementos básicos: por un lado el conocimiento necesario para representar historias en un ordenador, y por otro los procesos capaces de manipular dicho conocimiento para generar historias que satisfagan unos criterios preestablecidos de valor y novedad. La propuesta utiliza lógicas descriptivas y programación orientada a objetos para construir un armazón compuesto de un núcleo software y un repositorio extensible de componentes ontológicos reutilizables que encapsulan la semántica declarativa y operacional de los principales dominios relacionados con la

narración automática.

Los resultados obtenidos han sido la implementación de este armazón y de una aplicación de ejemplo como instancia concreta suya, que ilustra los aspectos más técnicos sobre cómo llevar a la práctica la metodología propuesta para el desarrollo de aplicaciones de narración automática.

La evaluación de esta propuesta y de sus resultados da lugar a una discusión sobre la expresividad que presenta esta solución frente a las demás propuestas revisadas previamente. También se discute lo que se ha averiguado acerca de la autoría narrativa computacional, la forma en que son presentadas y evaluadas las historias, y el alcance de la validez de dicha evaluación.

Finalmente se presentan las conclusiones de esta investigación, incluyendo algunas sugerencias sobre cuales han de ser las líneas de investigación futuras.

Palabras clave

Generación de Historias, Narratología Computacional, Ingeniería Ontológica, Lógicas Descriptivas, Creatividad Computacional

A mi esposa

Convenciones tipográficas

En este documento se utilizan las siguientes convenciones tipográficas:

- Las citas cortas aparecen en el texto *"entre comillas"*.
- Las citas largas aparecen...

  > En un párrafo aparte, con un sangrado mayor y un tamaño de letra menor.

- Los nombres propios de centros de investigación, empresas, obras de arte, sistemas o herramientas informáticas aparecen *en letra cursiva*. Lo mismo ocurre con aquellos términos extranjeros que aparecen ocasionalmente en el texto.
- Los nombres propios de conceptos, roles, individuos, clases, objetos u otros elementos software aparecen en letra Sans Serif.
- Los fragmentos de código o de texto que se muestra por pantalla aparecen...

```
En cuadros aparte como este.
```

Índice general

1. Introducción **1**
- 1.1. Definición de narración automática . . . 2
- 1.2. Aplicaciones de narración automática . . . 2
- 1.3. Problemas abiertos . . . 4
- 1.4. Objetivos . . . 6
  - 1.4.1. Autoría computacional genérica y configurable . . . 7
  - 1.4.2. Evaluación integrada en la aplicación . . . 7
  - 1.4.3. Validación de la propuesta . . . 8
- 1.5. Estructura del trabajo . . . 8

2. Revisión del trabajo previo **11**
- 2.1. Estudio del dominio . . . 11
  - 2.1.1. Narratología . . . 12
  - 2.1.2. Narratología computacional . . . 17
- 2.2. Proyectos de investigación en narración automática . . . 20
  - 2.2.1. Perspectivas . . . 20
  - 2.2.2. Objetivos . . . 22
  - 2.2.3. Materiales y métodos . . . 24
  - 2.2.4. Resultados . . . 35
  - 2.2.5. Estado de la cuestión . . . 44
- 2.3. Descripción de las herramientas . . . 44
  - 2.3.1. Ontologías y bases de conocimiento . . . 45
  - 2.3.2. Lógicas descriptivas . . . 49
  - 2.3.3. Creatividad y razonamiento computacional . . . 53
  - 2.3.4. Armazones orientados a objetos . . . 60

3. Armazón para narración automática **63**
- 3.1. Hipótesis . . . 63
- 3.2. Material y método de investigación . . . 64
- 3.3. Núcleo del armazón . . . 66
  - 3.3.1. *DLModel* . . . 66

3.3.2. *DLApplication* 68
3.4. Componentes básicos del armazón 70
3.4.1. *Knowledge* 75
3.4.2. *Space* 79
3.4.3. *Time* 82
3.4.4. *Simulation* 86
3.4.5. *Fabula* 93
3.4.6. *Discourse* 98
3.4.7. *Narration* 103
3.4.8. *AbstractStory* 104
3.5. Funcionamiento del armazón 109
3.6. Evaluación del armazón 114
3.6.1. Coherencia 116
3.6.2. Valor 116
3.6.3. Novedad 118
3.6.4. Calidad de presentación 121

4. Validación del armazón **123**
4.1. Componentes específicos de la aplicación 124
4.1.1. *Tale* 125
4.1.2. *Propp* 130
4.1.3. *KICBR* 134
4.2. Funcionamiento de la aplicación 138
4.3. Evaluación de la aplicación 142

5. Discusión **147**
5.1. Representación de historias 147
5.2. Generación de historias 150
5.3. Evaluación de historias 153
5.4. Historias resultantes 154

6. Conclusiones **161**
6.1. Genericidad y dependencia en autoría computacional 162
6.2. Evaluación automática con métricas abstractas 163
6.3. Validez delegada al modelo de evaluación 164
6.4. Trabajo futuro 164

A. Documentación complementaria **III**

B. English Translation **V**

C. Abreviaciones **XI**

D. Glosario xiii

Índice de Cuadros

2.1. Resumen del proyecto *Automatic Novel Writer* 21
2.2. Resumen del proyecto *Minstrel* 22
2.3. Resumen del proyecto *Brutus* 23
2.4. Resumen del proyecto *Defacto* 23
2.5. Resumen del proyecto *SWAN* 24
2.6. Resumen del proyecto *Joseph* 25
2.7. Resumen del proyecto *Rumelhart* 25
2.8. Resumen del proyecto *Mexica* 26
2.9. Resumen del proyecto *Tale-Spin* 26
2.10. Resumen del proyecto *Universe* 30
2.11. Historia generada mediante *Automatic Novel Writer* 36
2.12. Historia generada mediante *Tale-Spin* [Mee81] 36
2.13. Historia generada mediante *Joseph* [Lan97] 37
2.14. Historia generada mediante *Brutus* [BF99] 38
2.15. Historia generada mediante *Storybook* [CL02] 39
2.16. Historia generada mediante *Universe* [Leb87] 39
2.17. Historia generada mediante *Minstrel* [Tur92] 40
2.18. Historia generada mediante *Mexica* [PyPS01] 42
2.19. Resumen del proyecto *ISRST* 43
2.20. Lista de primitivas de la Teoría de la Dependencia Conceptual 48

3.1. Resumen de la API del componente *Knowledge* 78
3.2. Resumen de la API del componente *Space* 83
3.3. Resumen de la API del componente *Time* 86
3.4. Resumen de la API del componente *Simulation* 93
3.5. Resumen de la API del componente *Fabula* 98
3.6. Resumen de la API del componente *Discourse* 102
3.7. Resumen de la API del componente *Narration* 104
3.8. Resumen de la API del componente *AbstractStory* 109

4.1. Resumen de la API del componente *Tale* 129

4.2. Resumen de la API del componente *Propp* 134
4.3. Resumen de la API del componente *KICBR* 136
4.4. Historia del corpus incluido en *ProtoPropp* 142
4.5. Historia generada aleatoriamente en *ProtoPropp* 143
4.6. Historia generada con todos los componentes de *ProtoPropp* . 143
4.7. Resultados de aplicar las métricas formales 143
4.8. Instrucciones para los jueces del experimento 144
4.9. Frecuencias sobre la calidad lingüística 145
4.10. Estadísticos sobre la calidad lingüística 145
4.11. Frecuencias sobre la coherencia 145
4.12. Estadísticos sobre la coherencia 145
4.13. Frecuencias sobre el interés 146
4.14. Estadísticos sobre el interés 146
4.15. Frecuencias sobre el interés 146
4.16. Estadísticos sobre la originalidad 146

Índice de Figuras

2.1. La aplicación *ISRST* narrando una animación 41
2.2. Animación generada mediante el sistema *SWAN* 43

3.1. Diagrama de despliegue de una aplicación *DLApplication* . . . 69
3.2. Diagrama de clases de un componente según *DLApplication* . 70
3.3. Jerarquía de componentes básicos del armazón 74
3.4. Jerarquía conceptual del componente *Knowledge* 76
3.5. Jerarquía de roles del componente *Knowledge* 77
3.6. Jerarquía conceptual del componente *Space* 80
3.7. Jerarquía de roles del componente *Space* 81
3.8. Jerarquía conceptual del componente *Time* 84
3.9. Jerarquía de roles del componente *Time* 85
3.10. Jerarquía conceptual del componente *Simulation* (1/2) 88
3.11. Jerarquía conceptual del componente *Simulation* (2/2) 89
3.12. Jerarquía de roles del componente *Simulation* (1/2) 91
3.13. Jerarquía de roles del componente *Simulation* (2/2) 92
3.14. Jerarquía conceptual del componente *Fabula* 95
3.15. Jerarquía de roles del componente *Fabula* (1/2) 96
3.16. Jerarquía de roles del componente *Fabula* (2/2) 97
3.17. Jerarquía conceptual del componente *Discourse* 99
3.18. Jerarquía de roles del componente *Discourse* (1/2) 101
3.19. Jerarquía de roles del componente *Discourse* (2/2) 102
3.20. Jerarquía conceptual del componente *Narration* 103
3.21. Jerarquía de roles del componente *Narration* 105
3.22. Jerarquía conceptual del componente *AbstractStory* 106
3.23. Jerarquía de roles del componente *AbstractStory* (1/2) 107
3.24. Jerarquía de roles del componente *AbstractStory* (2/2) 108

4.1. Jerarquía de componentes específicos de *ProtoPropp* 124
4.2. Jerarquía conceptual del componente *Tale* (1/3) 126
4.3. Jerarquía conceptual del componente *Tale* (2/3) 127

4.4. Jerarquía conceptual del componente *Tale* (3/3) 128
4.5. Jerarquía conceptual del componente *Propp* (1/3) 131
4.6. Jerarquía conceptual del componente *Propp* (2/3) 132
4.7. Jerarquía conceptual del componente *Propp* (3/3) 133
4.8. Jerarquía conceptual del componente *KICBR* 135
4.9. Jerarquía de roles del componente *KICBR* 137
4.10. Estableciendo la solicitud inicial para *ProtoPropp* 138
4.11. Buscando una fábula en el corpus de *ProtoPropp* 139
4.12. Generando una fábula aleatoria en *ProtoPropp* 140
4.13. Generando una nueva fábula mediante *ProtoPropp* 141

Capítulo 1

Introducción

"¿Por donde debería comenzar, su Majestad?" preguntó.
"Comienza por el principio," dijo el Rey, solemne,
"y continúa hasta llegar al final: entonces para."
Lewis Carroll

Con el nacimiento de la Inteligencia Artificial (IA) como disciplina científica, los sistemas informáticos dejaron de ser vistos únicamente como herramientas de ingeniería para realizar cálculos pesados y almacenar grandes cantidades de datos. Varias décadas después la Informática se ha convertido en una disciplina al servicio de todas las demás, sirviendo incluso para modelar fenómenos y construir herramientas para los artistas, los psicólogos y muchos otros especialistas en el área de las Humanidades. Hoy día, gracias a la popularización de las nuevas tecnologías de la información, el ordenador se ha convertido en una herramienta potente y completa capaz de realizar una multitud de tareas, incluida una de las más complejas y características del ser humano: la narración de historias. Una narración que hoy día abarca toda clase de categorías y géneros, desde las obras clásicas de la literatura y el cine en formato digital hasta las más vanguardistas exhibiciones de realidad virtual, pasando por la animación infográfica, los efectos especiales, la dramatización robótica y los videojuegos.

El viejo anhelo de reproducir comportamiento inteligente en las máquinas se ha visto alimentado hoy día por la capacidad multimedia de los sistemas actuales, sugiriendo un enorme potencial artístico y educativo subyacente en las aplicaciones narrativas de la informática [Mur97]. La capacidad de automatizar tareas, tan propia de la computación, ha propiciado que desde hace más de tres décadas se hayan desarrollado tecnologías que intentan convertir a las máquinas en algo más que una herramienta para el narrador [Hir98]. Este es el caso de las aplicaciones de *narración automática*.

1.1. Definición de narración automática

Entendiendo la narración como un proceso en el cual un autor crea y transmite una historia al público, la narración automática es el proceso en el cual un sistema informático crea y transmite una historia al público sin que dicha historia sea conocida de antemano por ningún ser humano.

Las historias deben además cumplir unos requisitos de coherencia y satisfacer unos criterios mínimos sobre su valor y su novedad con respecto a otros historias generadas, además de poder ser representadas mediante recursos textuales, gráficos o de otro tipo de manera directa e inteligible frente a un público real.

1.2. Aplicaciones de narración automática

Las aplicaciones de narración automática tuvieron su auge en los años setenta y ochenta, cuando se llevaron a cabo numerosos estudios relacionados con la comprensión y la generación automática de historias que sentaron las bases para la investigación en esta materia. Tras años de desinterés por la cuestión, debido al fracaso de grandes proyectos universalistas y excesivamente ambiciosos en el campo de la IA, actualmente hay nuevas motivaciones en el ámbito científico y empresarial que han provocado un renacer de la investigación en narración automática. La principal motivación son las aplicaciones similares que hoy día están viviendo un gran auge, como los videojuegos narrativos. La potente industria del entretenimiento está muy interesada en beneficiarse de las tecnologías de narración automática para disminuir costes de producción. En segundo lugar hay nuevos enfoques, como los estadísticos o basados en la explotación de los recursos públicos de Internet, que podrían aplicarse para mejorar dichas tecnologías. En tercer lugar, se dan las condiciones técnicas adecuadas para que, tanto los nuevos enfoques como los antiguos, puedan ser implementados con éxito sin imponer requisitos software o hardware excepcionales. Esto es debido a que en los últimos años las máquinas de los usuarios han aumentado enormemente su capacidad de computación, sus prestaciones multimedia y la calidad de su conexión a la red.

El interés en la investigación multidisciplinar y creativa dentro de la Informática se justifica también por su mención explícita en la agenda investigadora de diversas instituciones oficiales, como la *National Academy of Sciences* americana [MBI03] o la *DG Information and Communication Technologies* [Med] europea.

En general la investigación en narración automática puede justificarse

desde dos perspectivas: la humanista y la empresarial [IH]. Desde la perspectiva *humanista* esta investigación es una forma de estudiar, a través del modelado y la simulación, la capacidad de expresión, el arte y la creatividad humana. Desde la perspectiva *empresarial*, la narración automática ofrece ventajas evidentes al reducir los costes de creación de nuevas historias, garantizando la reutilización y ampliación sostenible de los recursos narrativos de que disponga una empresa. El producto ofrecido, una aplicación de narración automática, será más valioso al poder ofrecer un mayor número de historias al público de las que ofrece una aplicación que solo contiene historias creadas por autores humanos.

Las historias narradas por una aplicación de narración automática son asunto de sus desarrolladores y pueden tratar sobre muchos y muy diversos temas. En este apartado se recogen algunas referencias significativas de proyectos de narración automática que han sido aplicados con éxito en diferentes dominios, para justificar este esfuerzo de investigación y demostrar el interés multidisciplinar existente en el desarrollo de estas aplicaciones.

En el terreno artístico se centran muchos de los proyectos enfocados al desarrollo de dramas interactivos [Bat92, MS03], siendo más notable la orientación lúdica hacia los videojuegos [Mag02] y otro software de entretenimiento de narración digital e interactiva [CCPL, Cra]. En el ámbito de la publicidad también existen empresas dedicadas a incorporar aplicaciones de narración automática en los sitios web de sus clientes [Zoe].

Por otro lado las aplicaciones educativas, y lo que se conoce actualmente como *juegos serios*, también se benefician de la tecnología de narración automática, como es el caso de los proyectos de entrenamiento militar [GI03], educación en Historia [RY04], educación infantil [Doy], museos virtuales [MOOK98], o simuladores sociales [Ext, eDr].

Finalmente hay avances incluso en el terreno de la terapia psicológica, donde la realidad virtual en combinación con las tecnologías de narración automática también están ayudando a curar ciertas patologías [Hug88, RHK+95, HRW+96].

El éxito de mucha de estas aplicaciones no se debe únicamente a las tecnologías de narración automática que incorporan, sino también a su adecuada integración con determinados simuladores de entornos virtuales complejos, habitantes inteligentes o potentes interfaces hombre-máquina de realidad virtual.

Todos estos ejemplos se irán analizando con detalle partiendo de proyectos reales que se recogen en la literatura, aunque hoy en día el campo de la Narración Digital está todavía dando sus primeros pasos y es de esperar que surjan nuevas aplicaciones a medida que la investigación va dando resultados.

1.3. Problemas abiertos

El desarrollo de aplicaciones de narración automática tiene muchos problemas comunes al desarrollo de sistemas basados en conocimiento, dado que normalmente este es el enfoque que se sigue a la hora de diseñar estas aplicaciones. El cuello de botella del desarrollo suele ser la adquisición de conocimiento. En el caso de la narración automática apenas existen corpus etiquetados con el conocimiento necesario para generar nuevas historias, ya que es muy difícil encontrar fuentes narrativas apropiadas de donde sacar todo ese conocimiento. La cantidad de conocimiento necesaria es muy vasta, relativa a muchos y muy diversos dominios, incluyendo dominios tan ambiguos como el del "sentido común" o tan complejos y difíciles de formalizar como el de la psicología. Además buena parte de conocimiento necesario para comprender una historia es implícito o debe ser inferido por el público a través de un proceso de interpretación complejo y lleno de matices que son los que precisamente dan valor a la historia.

Vinculado a la cuestión del conocimiento y su representación computacional hay otro problema: el proceso de generación de historias. Toda narración implica la construcción de uno o más modelos de simulación de mundos ficticios, modelos abstractos que a su vez deben ser convertidos en modelos computacionales de lo narrado, para ser representados en el ordenador junto al resto de la información referente al proceso narrativo. La generación de historias estará formada por un proceso o un conjunto de procesos de razonamiento que manipulan conocimiento declarativo y operacional heterogéneo. Por si esto no fuera suficiente la generación de historias es un proceso creativo, siendo la creatividad una de las capacidades más asombrosas y emblemáticas del pensamiento humano y su automatización objeto de un más que controvertido debate.

Con respecto a las aplicaciones de narración automática existentes hay que decir que la mayoría provienen, cronológica o conceptualmente, de la época dorada de la inteligencia artificial, por lo que desgraciadamente padecen los mismos problemas que condujeron a la decadencia de esta disciplina durante los años 80.

En primer lugar, los objetivos que suelen plantear son muy ambiciosos y poco concretos dado que intentan resolver a la vez demasiados problemas sin definición clara.

En segundo lugar, no se identifican los problemas abiertos en el estado del arte ni se compara el trabajo previo con el realizado en relación a las contribuciones que cada uno de ellos aporta a una supuesta causa común. En ocasiones se exploran soluciones nuevas con enfoques muy diferentes a las anteriores, evitando de forma injustificada la comparación con el trabajo

previo.

En tercer lugar muchas decisiones metodológicas y de diseño se toman *ad hoc*, sin basarse en una teoría sólida o un estudio empírico suficiente. Falta un buen marco de trabajo donde clasificar cada enfoque y entender sus ventajas e inconvenientes, siendo deseable contar con un sistema concreto para desarrollar aplicaciones de narración automática.

En cuarto lugar, para obtener resultados convincentes se hace necesario simplificar en exceso dichos problemas lo que contribuye a que los enfoques adoptados para resolverlos tiendan a ser obvios y poco realistas. Normalmente esta simplificación implica que la cantidad de contenido disponible para generar historias sea escasa y su variedad quede reducida a un sólo dominio temático muy limitado. La excesiva simplificación también conduce a acoplar la tecnología desarrollada con los problemas concretos que se utilizan para probar el funcionamiento de la aplicación, lo que hace que las soluciones obtenidas no puedan ser escaladas para resolver problemas de mayor tamaño o complejidad, ni generalizadas para resolver problemas similares.

En quinto lugar, la metodología y los resultados de la investigación no son evaluados correctamente, llegando incluso a no aparecer documentados en algunos trabajos. La evaluación, cuando existe, no suele usar métricas objetivas ni realizar experimentos suficientemente exhaustivos como para probar la validez de la solución propuesta. Los resultados tampoco se comparan con los resultados de otros proyectos previos.

En sexto y último lugar, muchas soluciones están ligadas a la implementación de aplicaciones de prueba que carecen de utilidad práctica y no responden a las necesidades de ningún cliente ni mercado real, por lo que no despiertan interés económico ni social. Las soluciones no se utilizan en más de una aplicación, a veces porque no son reusables, otras veces porque los autores no logran hacer efectiva dicha reusabilidad. Algunos trabajos directamente carecen de implementación y por tanto de resultados prácticos. Esto hace muy difícil extrapolar las conclusiones para hablar de la mejora, en términos de coste y beneficio, que las nuevas soluciones representan para el desarrollo de aplicaciones de narración automática.

Teniendo en cuenta estos problemas, son muchos los retos que el investigador en la materia puede plantearse. Por un lado hay problemas que pertenecen al ámbito de la Creatividad Computacional y que plantean el reto de averiguar qué representaciones computacionales de los elementos de una historia y qué procesos computacionales para generar historias válidas y novedosas a partir de dichos elementos son los más adecuados para cada aplicación de narración automática. Por otro lado hay problemas, dentro del ámbito de la Narratología Computacional, que solo se dan en ciertos tipos de narración. Por ejemplo, la narración interactiva presenta un dilema complejo [Pei04] al

intentar que tanto los autores como los interactores controlen la generación de la historia. Otro ejemplo es el de la narración adaptativa, que presenta problemas similares a los que se encuentran en aplicaciones de personalización de contenidos. Finalmente existe un tercer ejemplo, la adaptación multimodal, que presenta problemas pertenecientes al ámbito de la Interacción Hombre-Máquina. En principio toda clase de recursos multimedia pueden utilizarse para presentar historias al público, desde un volcado de datos en texto plano a los entornos virtuales tridimensionales más sofisticados visualmente, pasando por todas las distintas etapas de generación de lenguaje natural. A menudo es importante definir una capa de presentación independiente de la representación interna que el sistema utiliza para construir la historia. Esta capa de presentación, de nuevo para los casos extremos de la narración interactiva y multimodal, puede llegar a tener requisitos de funcionamiento en tiempo real, comprensión de las acciones que realiza el usuario o incluso de órdenes en lenguaje natural.

Como es obvio, solo algunos de estos retos son abordados en este trabajo, los cuales se definen como objetivos concretos en el siguiente apartado.

1.4. Objetivos

Todo trabajo de investigación tiene como propósito la adquisición y difusión de nuevo conocimiento. En el ámbito de esta tesis, se desea conocer algo más sobre la naturaleza y el funcionamiento de las aplicaciones de narración automática de cara a difundir las novedades entre los ingenieros que las desarrollan para facilitar su construcción y mejorar su calidad.

El propósito general de esta investigación es construir un armazón adecuado para el desarrollo de aplicaciones de narración automática. Se desea contribuir así en la proliferación de herramientas de utilidad para en este tipo de proyectos. Para ello el armazón que se propone ha de ser válido y eficaz, además de todo lo óptimo y eficiente que sea posible, en el cumplimiento de dos funciones distintas pero igual de importantes.

La primera función consiste en satisfacer a los desarrolladores de las aplicaciones en términos de productividad software. Esto significa reducir los costes de desarrollo, es decir, maximizar la funcionalidad de las aplicaciones al tiempo que se minimizan los recursos necesarios para programarlas y el tiempo de desarrollo. Las etapas de desarrollo cubiertas serán el diseño, la construcción y el mantenimiento de este tipo de software.

La segunda función consiste en satisfacer a los usuarios de las aplicaciones en términos de calidad final del producto. Dado el carácter relativo de los criterios de calidad que se aplican a este tipo de productos, los desarrolladores

deberán disponer de alguna métrica de evaluación para los resultados de sus aplicaciones y poder así obtener realimentación con la que ajustarlas, ya sea en tiempo de ejecución o durante su desarrollo. En las aplicaciones de narración automática la experiencia subjetiva del usuario en su papel de público es un factor determinante de la calidad, por lo que la métrica irá destinada a evaluar bien la utilidad y la novedad que cada usuario aprecia en los resultados o bien una aproximación de estas basada en heurísticas válidas objetivamente mensurables.

A través del método científico, que obliga a identificar los problemas y a analizar el trabajo previo primero, se concretan unos objetivos para este trabajo que se exponen a continuación.

1.4.1. Autoría computacional genérica y configurable

El objetivo principal de esta investigación es conseguir un armazón de autoría computacional que facilite el desarrollo de aplicaciones de narración automática. Este armazón debe permitir la implementación de algoritmos de generación automática de nuevas fábulas a partir de una solicitud inicial del usuario y una base de conocimiento previa. Además los algoritmos deben poder construir un discurso razonable para dicha fábula, elaborando contenidos sobre dominios de ficción genéricos y complejos, que finalmente puedan ser presentados al usuario en forma de historia inteligible que satisface su solicitud. Por último el armazón debe admitir parámetros que permitan establecer varias configuraciones posibles de su ejecución, para facilitar la realización de experimentos con los que estudiar, de forma conjunta o aislada, aquellas variables libres que forman parte del problema que se intenta resolver.

Con respecto a la cuestión de la Creatividad Computacional es necesario apuntar que este trabajo no busca participar en el debate filosófico correspondiente sino encontrar soluciones prácticas cuya creatividad no es necesariamente comparable con la creatividad humana, sino que se reduce a cumplir con unos criterios mínimos de validez y novedad previamente establecidos.

Las historias generadas serán ficticias y no estarán basadas en ninguna historia previa ni base de conocimiento con datos sobre sucesos reales.

1.4.2. Evaluación integrada en la aplicación

Otro objetivo del proyecto, orientado a facilitar la posterior validación de esta propuesta, es encontrar una métrica de evaluación apropiada para las aplicaciones desarrolladas. Esta métrica estará integrada de manera razonable en la aplicación final, como parte de su repertorio de herramientas o

incluso métodos de sus componentes, lo que permitirá estudiar fácilmente el rendimiento de la aplicación y sus distintas configuraciones.

1.4.3. Validación de la propuesta

Finalmente, es decisivo considerar un último objetivo consistente en probar que la integración de los algoritmos de generación y las métricas mencionadas en el apartado anterior en un armazón debidamente diseñado, construido y documentado es una solución adecuada para el desarrollo de aplicaciones de narración automática.

Ha de ser posible implementar y experimentar con un ejemplo concreto de aplicación de manera que se puedan obtener datos fiables con los que concluir que la propuesta de esta tesis doctoral es válida.

1.5. Estructura del trabajo

Esta memoria de tesis está estructurada en siete capítulos cuyo contenido se resume a continuación.

El capítulo 1 introduce los conceptos fundamentales de este trabajo, presentado los campos de aplicación de la narración automática y sus problemas desde el punto de vista de esta investigación, así como los objetivos científicos que se persiguen en esta investigación y la forma en que esta memoria está organizada.

El capítulo 2 ofrece en primer lugar una revisión del dominio de la narración automática, como parte de una disciplina más amplia llamada Narratología Computacional, identificando cuales son los aspectos comunes a cualquier forma narrativa. En segundo lugar se ofrece una descripción de las herramientas teóricas y prácticas que se mencionan a menudo en este trabajo, de modo que el lector conozca los aspectos técnicos necesarios para comprender el contenido de ciertos apartados del trabajo. En tercer lugar se presenta el estado de la cuestión en cuanto a sistemas de desarrollo de aplicaciones de narración automática, identificando y clasificando los proyectos más relevantes en el área, exponiendo con detalle el método y los resultados de cada uno de ellos, para su posterior discusión y comparación con el método y los resultados de esta tesis.

El capítulo 3 parte de la hipótesis de este trabajo y explica cual ha sido el material utilizado durante la realización de este proyecto, así como el método de investigación empleado para alcanzar los objetivos propuestos. En este capítulo se expone el grueso de la contribución de este trabajo al estado de la cuestión. Primero se diseña un armazón para desarrollar apli-

caciones de narración automática que permite organizar de manera modular el trabajo de los desarrolladores y evaluar de forma objetiva el rendimiento de las aplicaciones. Después se aportan todos los detalles sobre la construcción e implementación de dicho armazón basado en componentes ontológicos reutilizables, ideas surgidas a raíz de la investigación documentada en este trabajo.

El capítulo 4 presenta los resultados de esta investigación, la aplicación desarrollada a partir del armazón propuesto y su evaluación, lo que permite ilustrar el rendimiento y la utilidad real de la propuesta presentada en el capítulo anterior.

El capítulo 5 discute la metodología y los resultados obtenidos de este proyecto de investigación. Se analizan las ventajas y desventajas del armazón propuesto y se compara con la metodología y los resultados de los sistemas analizados en el capítulo de revisión del trabajo previo.

Por último, el capítulo 6 recoge las conclusiones de esta investigación, junto con unos comentarios finales sobre cual ha de ser el trabajo futuro en el campo de la narración automática a partir de ahora.

Capítulo 2

Revisión del trabajo previo

La Ciencia y el Arte pertenecen al mundo entero,
y ante ellas se desvanecen las barreras de las naciones.
Johann Wolfgang Von Goethe

En este capítulo se realiza una revisión lo más exhaustiva y profunda posible sobre el trabajo previo a esta investigación. En primer lugar se procede a realizar un estudio del dominio de la narración automática. En segundo lugar se revisan las herramientas mencionadas en el apartado 2.2 y en el capítulo 3. En tercer y último lugar se analizan los diferentes proyectos de narración automática que se encuentran en la literatura científica, describiendo sobretodo los objetivos, los materiales y los métodos de investigación que presentan cada uno de ellos.

2.1. Estudio del dominio

La narración es un fenómeno al que estamos acostumbrados. Vivimos rodeados de historias y según algunos investigadores es precisamente en forma narrativa como el ser humano estructura mentalmente sus experiencias vitales.

En este apartado se analiza la narrativa desde el punto de vista morfológico, estudiando principalmente el significado *inmediato* de la misma, la interpretación directa del mundo ficticio que se nos describe, sin entrar a analizar los aspectos de su significado *profundo*, como la simbología o las figuras retóricas que encierra el relato. Se estudiarán los aspectos teóricos más importantes de la narración, presentes en todas las categorías narrativas como son la literatura, el teatro o el cine.

En primer lugar se repasan los conceptos fundamentales de la Narratología, para pasar después a estudiar cómo se implementan dichos conceptos en

términos computacionales.

2.1.1. Narratología

La Narratología es el estudio de la narrativa, de su estructura y de la forma en que esta es generada y comprendida por el ser humano. Aunque en ocasiones pueda ser vista como un caso particular en el estudio de la Lingüística o la Semiótica, dadas las inevitables dependencias que existen entre el análisis de la Lengua y el de la Literatura, hoy día esta disciplina posee entidad propia, vinculada estrechamente con otras como las Ciencias Cognitivas y Sociales, pero única en cuanto a su planteamiento y desafío particular.

La narrativa, especialmente las obras de ficción, es un artefacto complejo dada la dualidad que existe en los propósitos que, tanto el autor como el público, tienen con respecto al acto de la narración. En la dimensión racional de la narración el autor puede estar tratando de comunicar un mensaje, instruir al público según una determinada doctrina o simplemente obtener algún beneficio a cambio de satisfacer necesidades afectivas del público (deleite, entretenimiento, distracción, etc.). En la dimensión irracional de la narración el público trata de disfrutar experimentando distintas emociones reales, que varían según el contenido y la forma de lo narrado, cuyo carácter y nivel de tensión pueden ser manipulados por el autor en virtud de una serie de técnicas narrativas (misterio, suspense, sorpresa, etc.). Ambas dimensiones, como ocurre en muchos aspectos de la vida humana, están profundamente interrelacionadas y aunque en algunas historias van de la mano, en otras aparecen completamente enfrentadas, como es el caso de cualquier documental formativo y al mismo tiempo aburrido o cualquier comedia divertida y a la vez frívola.

Según el diccionario [RAE] una historia es la relación de cualquier aventura o suceso, ya sea un acontecimiento real o se trate de una invención.

Las historias *valiosas* se distinguen por ser capaces, no sólo de transmitir esa información, sino de involucrar emocionalmente al público y hacer que este se sienta identificado con lo que ocurre en ellas. Ese "valor" de una historia, además de potenciarse con las habilidades de un buen narrador y una buena forma de presentar la historia, está directamente relacionado con la calidad del argumento y de los personajes que intervienen en ella.

Fábula, discurso y presentación

En el libro *Story and Discourse* de Seymour Chatman [Cha86] se toman ideas de varios estudios clásicos de la narrativa para presentar una concep-

ción dualista y estructuralista de la literatura. Según Chatman cualquier forma narrativa se compone de dos elementos fundamentales: la *historia* propiamente dicha y su *discurso*. Esta distinción persigue separar la estructura narrativa de cualquiera de sus manifestaciones.

Aunque la terminología es aún un problema abierto en Narratología, en este trabajo interesa principalmente definir de una forma clara los conceptos que hay detrás de los términos, por lo que simplemente se utilizan las palabras que resulten más claras para entender los fundamentales narratológicos de en esta investigación.

Por ejemplo, los formalistas rusos también distinguen entre la *fábula*, los sucesos que van a ser relatados, y la *trama*, los sucesos según el orden del relato; pero utilizan para referirse a estos conceptos los términos rusos *fabula* y *sjuzet*.

La teoría fundamental del arte dramático también distingue entre estos dos elementos. Aparece por primera vez en las obras de Aristóteles [Ari74]. Para Aristóteles, la imitación de acciones en el mundo real (*praxis*), forma un argumento (*logos*), del cual se seleccionan (y posiblemente reordenan) los sucesos o unidades fundamentales que forman la trama (*mythos*).

Janet Murray describe también esa doble función del autor y narrador de una historia, por un lado la capacidad de concentrar lo interesante en unos pocos sucesos, el contenido de la historia, y por otro el de expresar esos momentos de la manera adecuada, el discurso narrativo:

> It takes dramatic compression the technique of abstracting complicated patterns of human experience into sharply focused moments. And to make those moments worth paying attention to, they must be shaped by an author's experience of the world to give them particularity and emotional expressiveness. [Hir98]

Chris Crawford [Cra05], de una manera distinta, insiste también en esa idea, llegando a afirmar que la esencia de una historia no entiende de espacio ni tiempo, sino que es una estructura de más alto nivel basada en la causalidad. Esto quiere decir que no es exactamente lo mismo un personaje o un evento narrativo, que las personas o el conjunto de sucesos del mundo ficticio en que estos se manifiestan.

Según Chatman, la historia es una cadena de *eventos* y todas las *entidades*[1] relacionadas con esos eventos que existen a nivel conceptual antes de ejecutar la narración. Las entidades pueden ser personajes, escenarios o

[1]En la traducción española de *Story and Discourse* [Cha86] los términos ingleses *event* y *existent* se traducen por *suceso* y *existente* respectivamente. En este trabajo se ha optado por términos más utilizados en informática, como son *evento* y *entidad*.

accesorios. Los eventos pueden ser acciones, si son causados por personajes, o sucesos si no lo son.

El discurso, en cambio, es la expresión real de esa historia, compuesto por la *manifestación*, el medio a través del cual se comunica el contenido, y la *transmisión*, las formas narrativas que se emplean en esa comunicación: tiempo, voz, punto de vista, etc. Generar un discurso no consiste simplemente en elegir las palabras adecuadas para narrar la historia. Lo más importante es establecer un *orden* en los eventos que van a narrarse, decidir la *perspectiva* en que van a ser narrados –primera, segunda o tercera persona– y la *voz* que va a emplearse en dicha narración –dependiente de la relación que se establezca entre el autor, el narrador y el público–. Fenómenos narrativos tan habituales como las anacronías (prolepsis o analepsis) o las elipsis se producen gracias a estos parámetros que permiten cambiar el discurso con respecto a lo que está siendo narrado.

Profundizando en la estructura de una historia, los eventos se dividen en *acciones* y *acontecimientos*. Cada acción tiene asociada un personaje que la realiza, mientras que los acontecimientos no tienen sujeto, como ocurre con los fenómenos climáticos.

No todos los teóricos proponen esa división tan clara entre fábula y discurso. Algunos, como Robert McKee [McK97], experto en la producción de guiones cinematográficos, proponen un enfoque bastante más práctico.McKee combina los conceptos de historia y discurso estructurando la historia en forma de árbol, con nodos bien diferenciados en contenido pero también ordenados según el criterio del discurso y la propia cronología de la película.

La raíz del árbol es la *historia*, que se descompone en *actos*, que a su vez se descomponen en *secuencias*, que se descomponen en *escenas* compuestas finalmente por los que McKee considera las hojas del árbol: los *golpes de efecto* (en inglés *beats*). Estos últimos son los eventos más simples posibles (atómicos) que se producen en el mundo ficticio y que contienen *significado narrativo*. El resto de nodos del árbol una historia tiene un *valor* narrativo de más alto nivel que está en juego y dicho valor lleva siempre una *carga*, positiva o negativa. Cuando se producen suficientes golpes de efecto la carga de una escena varía hasta producirse un cambio de signo, siendo dicho cambio lo que produce que una escena concluya y comience la siguiente. De la misma forma al concluir una escena, secuencia o acto la carga del correspondiente nodo superior del árbol varía y si la carga cambia de signo se produce un cambio de secuencia, acto o el fin de la historia.

Syd Field [Fie98] también ordena los niveles narrativos en una jerarquía similar consistente únicamente en historia, actos, escenas y golpes de efecto.

En este trabajo usaremos tres términos básicos para referirnos a las distintas componentes de una historia, similares a los que propone Mieke Bal

[Bal98]. En primer lugar se denomina *fábula* al conjunto de entidades y eventos del universo ficticio que el autor crea para dar cuerpo a una historia. En segundo lugar se llama *discurso* a la estructura que el autor genera al seleccionar algunas entidades y hechos de la fábula y ordenarlos de acuerdo a como va a narrarlos. Finalmente se denomina *presentación* a la forma en que el narrador transmite el discurso de una fábula, también llamada la *pseudohistoria*, al público a través de un medio narrativo concreto, haciendo realidad una historia concreta.

El autor de una historia es quien decide el propósito de la misma. Además existen tres funciones distintas de narración que pueden realizarse por uno o varios narradores diferentes. La primera función es la creación de la fábula, que en la práctica siempre conlleva una primera limitación del contenido de la misma, ya que no es posible representar exhaustivamente la infinita complejidad de un universo ficticio. La segunda función es la construcción del discurso, produciendo la pseudohistoria al crear el discurso para la fábula generada. La tercera función es la realización de la presentación en un medio narrativo concreto, dando lugar a una verdadera historia.

Argumento

Desde tiempos de Aristóteles, el argumento de una historia se ha visto como el relato de una sucesión de hechos que hace énfasis en la *causalidad* [Cha86].

> A plot is... a narrative of events, the emphasis falling on causality. "The king died and then the queen died" is a story. "The king died and then the queen died of grief" is a plot. [For41]

La idea de definir una historia como una estructura compuesta de escenarios, personajes, objetos y toda una serie de piezas que forman el argumento fue expuesta por primera vez por el formalista ruso Vladimir Propp, en su obra *Morphology of the Folktale* [Pro68]. La intención de Propp era identificar la morfología de los *cuentos maravillosos*, una categoría importante dentro del género de los cuentos tradicionales rusos, partiendo del análisis de un corpus con cien ejemplares. Este corpus es la colección de Alekandr Nikoalevich Afanasiev, una recopilación de lo más significativo del folklore literario ruso.

En la morfología proppiana se presentan una serie de "funciones del personaje" con las que se puede representar la estructura narrativa de cualquier cuento maravilloso del corpus que analizó Propp. El concepto de *función de personaje* hace referencia a una serie de acciones que realiza un personaje

que de alguna manera están marcadas por la función que realizan a nivel narrativo dentro del argumento.

> Function is understood as an act of a character, defined from the point of view of its significance for the course of the action. [Pro68]

Estas funciones del personaje son elementos *invariantes* del cuento, *independientes* de los individuos concretos que toman parte en ellas y de las acciones concretas que estos realizan para llevarlas a cabo. Algunos ejemplos son las maldades que comete el villano (asesinato, secuestro, maltrato del héroe o de su familia, etc.) o las formas en la que un cuento concluye (boda, ascensión al trono del protagonista, recompensa económica, etc.). Propp sugiere que la extensa lista de funciones que él propone es un *conjunto reducido* de átomos fundamentales con los que pueden componerse todos los cuentos maravillosos, dando incluso una fórmula para representar todas las posibles combinaciones.

Otras ideas sobre el argumento se centran en la definición de los distintos tipos de conflicto que pueden darse en una historia, siendo el centro de gravedad de la misma. McKee [McK97] sugiere que existen tres tipos: los *cósmicos*, como los que se producen entre las fuerzas del Bien y del Mal, los *sociales*, como los que generan los triángulos amorosos, y los *personales*, como los traumas infantiles que dejado marcados para siempre a los protagonistas de una historia.

Han sido varios los autores que han tratado de englobar en una sola teoría todos los argumentos básicos que puede tener una historia. Murray [Mur97] hace referencia a esta búsqueda incansable de los narratólogos que ha dado frutos muy dispares, señalando que Jorge Luis Borges cree que existen menos de doce argumentos básicos diferentes, Ronald Tobías describe veinte y Rudyard Kipling llega hasta los sesenta y nueve.

Personajes

Los personajes son el otro elemento fundamental de una historia. Aunque Aristóteles en *La Poética* [Ari74] considerase la acción el ingrediente narrativo más importante, por encima incluso de los personajes, puede decirse que por norma en todas las historias aparecen personajes y su función, como hemos visto antes, es siempre muy relevante para el argumento.

Tal es así que McKee [McK97] defiende que el argumento y los personajes no son más que las dos caras de un mismo fenómeno narrativo, por lo que carece de sentido hablar como si una de ellas pudiera desaparecer de la ecuación o figurar en subordinación a la otra.

Según Chatman, las entidades de una historia se dividen en *personajes* y elementos del *escenario*. En la mayoría de las historias los personajes se distinguen por ser seres humanos u otro tipo de criaturas antropomorfas cuyas acciones forman parte de los eventos clave del argumento. Los elementos del escenario son los lugares y los objetos, que en ocasiones también pueden tener una función importante pero siempre mucho menos relevante para desarrollar la historia que la función de los personajes.

2.1.2. Narratología computacional

La Narratología Computacional se ocupa de una parcela concreta de la Narratología, estrechamente vinculada con la Informática: la representación y el procesamiento de la narrativa en términos computables, siendo el desarrollo de aplicaciones capaces de narrar historias automáticamente una de las motivaciones para dicho estudio.

Historia

Podemos decir que esta disciplina nace en los años 70 y 80, con el interés de Ciencia Cognitiva por reproducir la manera en que el ser humano comprende y recuerda enseñanzas en forma de historias. El trabajo en este área lo impulsó el grupo de Roger Schank en la universidad de *Yale*, que inicialmente se dedicaba a estudiar, desde un punto de vista psicológico, las estructuras y los procesos de la mente relacionados con el lenguaje natural.

Durante el desarrollo de programas como *SAM* [Cul81] o *PAM* [Wil81] se empezó a gestar una teoría sobre la narración de historias que se centraba en los personajes y las diferentes formas en que estos pueden perseguir sus objetivos, además de proponerse otros modelos para la memoria narrativa y el recuerdo [Kol84]. Pronto surgió la idea de usar la misma teoría en el sentido inverso: la generación de historias, lo que llevaría a la construcción del clásico *Tale-Spin* [Mee76, Mee81] y de otros sistemas emblemáticos de resolución de problemas y modelos de creatividad.

Desgraciadamente este tipo de proyectos tan relacionados con las humanidades y las ciencias experimentales fueron abandonados durante el invierno de la IA en favor de otros con objetivos más prácticos y aplicaciones más provechosas a corto y medio plazo. Con las excepciones de Mueller [Mue87] y Turner [Tur92] la investigación sobre narración automática quedó congelada hasta finales de los años 90.

A partir de 1999 resurge el interés en este tema gracias a proyectos de investigación multidisciplinares sobre Inteligencia Narrativa [MS99] y a la

decisiva integración de las nuevas tecnologías en el estudio de la Narratología [Mei].

En el taller AAAI sobre Inteligencia Narrativa celebrado en 1999 [MS99] se identificaron cinco áreas de estudio posibles dentro de esta nueva disciplina. En primer lugar la formalización teórica de la narrativa; en segundo lugar el desarrollo de sistemas de generación de historias; en tercer lugar los sistemas de explotación de la inteligencia narrativa humana; en cuarto lugar los sistemas de narración digital e interactiva; y en quinto lugar los agentes computacionales que poseen inteligencia narrativa o forman parte de un sistema con inteligencia narrativa.

El área de estudio que más interesan desde el punto de vista de este trabajo es la segunda, que podríamos llamar "Inteligencia Narrativa Artificial", aunque también se tengan presentes cuestiones importantes del tercer área, como la utilidad de las herramientas de autoría.

El medio natural donde estudiar Narratología Computacional es el medio digital, por lo que se denomina narración digital al relato que utiliza principalmente los procesos y los recursos de un sistema informático para su presentación. Este medio resulta especialmente idóneo para la narración automática, gracias a la capacidad de computación actual de los ordenadores. Las tecnologías de presentación son precisamente otro área distinta que se tendrá en cuenta por estar relacionada con la generación de historias.

Enfoques

La teoría fundamental del arte dramático que mencionábamos en el apartado 3.4.6 ha servido para inspirar varios libros y proyectos de narración automática [Lau91, Sgo99]. Aunque es probablemente la morfología proppiana la teoría más utilizadas en el desarrollo de aplicaciones de narración automática, como se podrá comprobar en el apartado 2.2.

Aunque la teoría de Propp se creó para analizar historias, muchos trabajos la toman como base para construir generadores de historias, básicamente considerando las funciones de personaje como los bloques básicos de construcción de un argumento. Esto es así porque no resulta difícil implementar el procesamiento algorítmico de su fórmula semiótica en un sistema informático, siendo fácil implementar por ejemplo un generador pseudoaleatorio de secuencias de funciones de personaje para cuentos maravillosos [SLTW].

No obstante es importante señalar que existen otros trabajos menos populares pero que analizan de forma más profunda y completa los contenidos de las historias, sugiriendo también algoritmos de generación similares como es el caso de la travesía del héroe arquetípico de Joseph Campbell [Cam68].

Tampoco faltan opiniones en contra de este tipo de enfoques "clásicos".

El argumento más convincente defiende que la narrativa moderna es incomparable en complejidad a la de los cuentos maravillosos que analizaba Propp y que por tanto requiere de nuevas teorías para ser generada.

Retos

Dentro de los sistemas de información, este trabajo trata sobre la construcción de sistemas basados en conocimiento. Todos estos sistemas presentan retos comunes en su construcción, como el alto coste de adquisición del conocimiento y la dificultad de su reutilización, en parte a causa del llamado *problema de interacción*, o dependencia entre el formato en que se representa el conocimiento y el algoritmo con el que dicho conocimiento será procesado.

La representación y el procesamiento de conocimiento narrativo presentan además un serio problema de escala a la hora de desarrollar aplicaciones de narración automática. Las historias tratan sobre mundos ficticios, pero esos mundos reflejan de alguna forma la realidad, y la realidad puede modelarse en infinidad de formas diferentes, atendiendo a infinidad de dominios de conocimiento distintos y arbitrariamente complejos, siendo precisamente los más complejos aquellos que más interesan para modelar los aspectos enciclopédicos, abstractos y recursivos de la psicología humana que se esconden detrás de una historia.

Es por ello que la primera precaución que hay que tomar al acercarse a este campo de estudio es acotar lo que se desea representar según se desea procesar en función de lo que se desea conseguir con todo ello. El espacio y el tiempo, misterios insondables que gobiernan nuestra existencia, deberán ser reducidos a representaciones abarcables, como los grafos de localizaciones o las líneas temporales.

Los sucesos, elementos fundamentales en la constitución de una historia, deberán articularse en función de dicha representación espacial y temporal escogida.

El mundo ficticio, contexto de los sucesos antes mencionados, se convertirá entonces en una entidad compleja cuyas partes se distribuyen en el espacio y pasan secuencialmente de un estado a otro a lo largo del tiempo.

Todos estos problemas, la representación de eventos en forma de secuencia de estados o de secuencia de transiciones partiendo de un estado inicial y deseando alcanzar uno final, son considerados problemas clásicos de la IA cuya solución depende en última instancia del espacio en memoria y el tiempo de ejecución disponibles para los procesos que se desean ejecutar, ya sean estos de acceso, modificación, creación o destrucción de la información contenida en el modelo del mundo.

Sin ánimo de buscar soluciones universales, este trabajo se centra en con-

seguir resultados concretos que sean de utilidad para el desarrollo de aplicaciones de narración automática.

2.2. Proyectos de investigación en narración automática

En este apartado de estudian los proyectos anteriores de investigación y desarrollo de aplicaciones de narración automática que aparecen documentados en la literatura científica.

Algunos de estos proyectos obtuvieron como resultados sistemas o armazones relativamente genéricos con los que se implementaron una o más aplicaciones de narración automática, mientras que otros se dedicaron solamente trabajar en modelos teóricos o no llegaron a implementar ningún prototipo por verse interrumpidos o reorientados hacia objetivos distintos. En cualquier caso, los aspectos que interesa estudiar son cinco: la perspectiva sobre el problema de la que parten cada uno de ellos, los objetivos que persiguen, el material y método de investigación utilizado, los resultados obtenidos y finalmente sus contribuciones al estado de la cuestión que aún estén vigentes hoy día.

Como guía para recorrer adecuadamente este apartado se ofrecen, en forma de cuadros ilustrativos, varios resúmenes con la descripción básica de aquellos proyectos que resultan más relevantes para esta revisión. Se pueden encontrar más detalles sobre estos proyectos de investigación en las referencias originales que se citan en este trabajo.

2.2.1. Perspectivas

Los proyectos de narración automática, según su diversas perspectivas, pueden dividirse en dos grandes grupos: los *pioneros* y los *contemporáneos*.

Los proyectos pioneros buscaban la primicia histórica de obtener resultados convincentes en el área de la narración automática. El ejemplo más evidente es *Automatic Novel Writer* [KAB+73], descrito en el Cuadro 2.1. Este proyecto dio lugar a la primera aplicación recogida en la literatura científica capaz de generar historias en lenguaje natural razonablemente largas, lo que se consideraba un paso decisivo hacia la generación automática de "novelas".

A medida que fueron apareciendo más aplicaciones de narración automática, los objetivos se fueron refinando, aunque estaban siempre orientados a mejorar el valor subjetivo que el público podía hallar en las historias que dichas aplicaciones eran capaces de generar.

Objetivos: Narración automática **Material y método:** Morfología proppiana, simulación basada en reglas y redes semánticas, gramática lingüística **Resultados:** Generador de historias de misteriosos asesinatos

Cuadro 2.1: Resumen del proyecto *Automatic Novel Writer*

El objetivo básico propuesto por *Automatic Novel Writer* es válido para cualquier aplicación de este tipo: ser capaz de generar al menos el registro de una simulación coherente. Sobre esta base se añaden otras características, que posteriormente también se incluyeron en la mayoría de proyectos que se revisan en este apartado. Estas características son: la posibilidad de que el usuario introduzca una solicitud inicial al comienzo de la ejecución, en este caso el nombre de los personajes y una descripción de la situación inicial; el comportamiento de caja negra del sistema, generando una única salida al final de todo el proceso de generación; y finalmente la presentación, sencilla pero inteligible, de la historia resultante ante el público, habitualmente en forma de texto escrito en un lenguaje natural muy simplificado.

Los proyectos contemporáneos, junto con algunas excepciones de la década de los 80, estudian más bien la viabilidad de diferentes materiales y métodos de investigación para desarrollar aplicaciones de narración automática, no necesariamente dedicando mucho esfuerzo a desarrollarlos lo más completas posibles. Obviamente los resultados son siempre importantes, porque permiten evaluar el trabajo realizado, pero hoy día empieza a quedar claro que la calidad de los resultados es muy controvertida de evaluar debido a la variedad de dominios de conocimiento que pueden considerarse o no a la hora de construir una historia y la variedad de técnicas de presentación que pueden utilizarse para mostrar dicha historia al público. Las tendencias actuales incitan más al estudio de la autoría computacional, ya sea modelando este como un sistema automático capaz de generar historias o como una herramienta inteligente que asiste al autor humano en el proceso narrativo.

Minstrel [Tur92], descrito en el Cuadro 2.2, es un proyecto peculiar en el sentido de que parte de un planteamiento pionero, el sueño de generar historias, pero acaba con uno más contemporáneo, experimentar con un modelo general de la creatividad humana. Al término del proyecto se argumenta que el sistema trata de imitar los procesos que ocurren en la mente de un trovador o cuentacuentos (en inglés *minstrel*), procesos creativos extrapolables a la resolución de problemas en otros dominios.

Un nuevo punto de vista que aparece en proyectos recientes es la idea de hacer Arqueología Software, intentar reconstruir aplicaciones software desa-

Objetivos: Narración automática y modelo de creatividad **Material y método:** KI-CBR y algoritmos creativos para dar forma a redes semánticas **Resultados:** Generador de historias cortas en lenguaje natural sobre el Rey Arturo y los Caballeros de la Mesa Redonda

Cuadro 2.2: Resumen del proyecto *Minstrel*

parecidas u obsoletas usando tecnologías actuales con el propósito de estudiar lo que dichas tecnologías pueden aportar a los modelos clásicos, al tiempo que hacen más accesible dichos modelos para los investigadores actuales [PG06b, SJMM07].

2.2.2. Objetivos

Los objetivos científicos de este tipo de investigación han variado poco con respecto a los objetivos que propusieron hace décadas los proyectos pioneros. Estos trataban de diseñar un sistema o armazón de generación de historias, desarrollando una aplicación con la que poder evaluar de manera empírica o en ocasiones informal el valor narrativo de unas cuantas historias generadas.

Continuando con la tradición de esta ambición pionera, los proyectos se fueron centrando en uno o varios aspectos concretos del proceso de generación de historias, como por ejemplo en hacer más creíble el comportamiento de los personajes en las historias o mejorar las técnicas de presentación de las mismas ante el público. En muchos de estos casos en los que la presentación es un objetivo importante, se trata de mejorar la generación de lenguaje natural, tomando la aplicación de narración automática como caso práctico. Sin embargo estos proyectos a menudo también proponen teorías interesantes sobre la generación del discurso o incluso la fábula de una historia, probablemente debido a la estrecha relación existente entre el lenguaje y la narración, como ya se comentó en el apartado 2.1.1.

Brutus [BF99], descrito en el Cuadro 2.3, es un ejemplo de proyecto que plantea un objetivo distinto al de la generación automática de nuevas fábulas. El objetivo principal de este proyecto es crear una aplicación generadora de *versiones distintas* de una misma historia creada previamente por un autor humano, modificando ligeramente el discurso y de manera más profunda la presentación según la solicitud inicial del usuario. Curiosamente los autores defienden abiertamente la teoría de que no es posible construir una aplicación capaz de generar historias, siendo su sistema un ejemplo de que se puede simular el resultado pero no el proceso de la verdadera creatividad humana.

Objetivos: Calidad de la presentación lingüística en narración automática **Material y método:** FLEX (dialecto Prolog) para implementar reglas de transformación y gramáticas de historias **Resultados:** Aplicación generadora de versiones distintas de una historia sobre traición en el mundo académico en lengua inglesa

Cuadro 2.3: Resumen del proyecto *Brutus*

Otro posible objetivo, sugerido en el apartado anterior, es el de modelar la inteligencia o la creatividad humanas, ya sea utilizando la narración como caso práctico o como área concreta de estudio. En el caso del modelo creativo de *Minstrel*, la narración es el principal caso práctico, aunque el modelo fue posteriormente aplicado a dominios distintos. Algunos de los proyectos que se plantean este objetivo pueden ir buscando nuevas formas de asistir a las personas en el proceso narrativo, considerando inviable o poco interesante la posibilidad de llegar a automatizar semejantes actividades.

Finalmente, aunque la mayoría de los proyectos coincidan en el planteamiento general, los objetivos se vuelven cada vez más concisos, existiendo proyectos que por ejemplo se limitan al estudio de la narración interactiva o de la narración multimodal. Como ejemplo de narración interactiva se puede mencionar *Defacto* [Sgo99], descrito en el Cuadro 2.4, y como ejemplo de narración multimodal, el sistema *SWAN* [LZ02], descrito en el Cuadro 2.5.

Objetivos: Dirección automática de narración interactiva **Material y método:** Teoría Aristotélica implementada mediante lógica de primer órden en un aplicación Java y VRML **Resultados:** Aplicación de narración interactiva a modo de juego en un navegador web

Cuadro 2.4: Resumen del proyecto *Defacto*

Desde el punto de vista de este trabajo, todos pueden aportar ideas y herramientas útiles, aunque sin embargo es importante aclarar que los esfuerzos de revisión se concentran únicamente en aquellos proyectos que tienen suficientes objetivos en común con este trabajo como para poder ser comparables en términos de materiales y métodos utilizados en la investigación.

Objetivos: Narración automática de descripciones en lenguaje natural mediante animaciones **Material y método:** Planificación y gramáticas de historias en cinco niveles distintos **Resultados:** Generador de animaciones 3D mostradas en televisión a partir de descripciones en chino

Cuadro 2.5: Resumen del proyecto *SWAN*

2.2.3. Materiales y métodos

En este apartado se estudian los materiales y métodos de investigación utilizados en los proyectos que están siendo revisados.

Para la realización de este estudio se parte de la literatura científica disponible, como es el caso de los fondos bibliográficos de la Universidad Complutense de Madrid, especialmente aquellos libros sobre Narratología e Inteligencia Artificial que se encuentran en las Bibliotecas de Informática Ciencias de la Información, Filología, Filosofía y Biblioteconomía.

De especial importancia para la revisión son los artículos sobre temas específicos de narración automática recogidos en actas de congresos europeos (*International Conference on Virtual Storytelling* y *Technologies for Interactive Digital Storytelling and Entertainment*), talleres o simposios internacionales (*Artificial Intelligence and Interactive Digital Entertainment* y *AAAI 2007 Fall Symposium on Intelligent Narrative Technologies*) y en las memorias de proyectos de investigación anteriores [Pei04]. La mayor parte de estos documentos están disponibles en Internet y sólo en algunos casos a través de las distintas redes de bibliotecas universitarias, como es el caso de las numerosas publicaciones electrónicas a las que la Facultad de Informática está suscrita, especialmente *Lecture Notes in Computer Science* y *Lecture Notes in Artificial Intelligence*.

En situaciones muy puntuales este material se ha visto complementado con información proporcionada por otros investigadores de manera informal, en persona o a través de la web y del correo electrónico.

No es habitual encontrar disponible la documentación, la versión ejecutable y el código fuente de los proyectos de narración automática, aunque sí existen algunas herramientas de libre distribución y algunos investigadores dispuestos a facilitar más información a sus compañeros de la que puede encontrarse publicada.

El material básico para cualquier investigación son las herramientas informáticas para documentar el proyecto como LaTeX [pro], el sistema de edición de textos que se ha utilizado para preparar este documento, o *ArgoUML* [Tig],

la herramienta de diseño de diagramas y modelado de software.

Para permitir una difusión libre y segura del código generado se ha utilizado la licencia LGPL [GNU]. En cuanto a materiales más específicos de este tema de investigación, interesa conocer qué modelos y sistemas software se utilizan como herramientas en cada proyecto para representar y procesar la información de una aplicación de narración automática. En cuanto a metodología interesa conocer de qué forma se realiza el diseño, la implementación y la evaluación de dicha aplicación de narración automática.

Materiales de representación

A continuación se analiza el material utilizado para representar el conocimiento en los distintos proyectos revisados, tanto los modelos como las sistemas software que los implementan.

Muchos sistemas utilizan un modelo narratológico popular y lo transladan directamente a una representación computacional. *Automatic Novel Writer*, *Joseph* [Lan97], el sistema de *Rumelhart* [Rum75] y muchos otros están basados en la morfología proppiana y utilizan estas teorías para generar sus historias, a pesar de que originalmente fueran concebidas como teorías de análisis narrativo. *Joseph* y el sistema creado por *Rumelhart* son descritos brevemente en los cuadros 2.6 y 2.7, respectivamente.

Objetivos: Narración automática **Material y método:** Morfología proppiana, arquitectura modular: un modelo del mundo, predicados temporales, gramática para la historia e intérprete de dicha gramática **Resultados:** Generador de historias cortas en inglés

Cuadro 2.6: Resumen del proyecto *Joseph*

Objetivos: Narración automática **Material y método:** Morfología proppiana y generación gramatical de la historia **Resultados:** Generador de historias cortas en inglés

Cuadro 2.7: Resumen del proyecto *Rumelhart*

Normalmente se utilizan formalismos lógicos para representar el conocimiento y la estructura narrativa de las historias, siendo otra opción menos habitual emplear modelos matemáticos o estadísticos sencillos.

A continuación, el mundo ficticio se puede modelar de acuerdo a muchos dominios diferentes. *Mexica* [PyPS01], por ejemplo, modela más dominios de la simulación de los que incluye *Tale-Spin*, concretamente las relaciones afectivas entre los personajes de la historia. Sin embargo mientras que las relaciones afectivas de *Mexica* son obra de los propios autores del proyecto, *Tale-Spin* usa ideas tomadas de la Teoría de la Dependencia Conceptual para modelar las acciones de los personajes. Esta teoría se describe brevemente en el apartado 2.3.1, y las características de los sistemas *Mexica* y *Tale-Spin* se resumen en los cuadros 2.8 y 2.9, respectivamente.

Objetivos: Narración automática y modelo de creatividad del autor **Material y método:** Ciclos iterativos de actividad y reflexión sobre esqueletos de historias, implementación en Pascal **Resultados:** Generador de historias en inglés sobre los mexica

Cuadro 2.8: Resumen del proyecto *Mexica*

Objetivos: Narración automática **Material y método:** Teoría de la Dependencia Conceptual y planificación en *Lisp* **Resultados:** Generador de historias cortas en inglés sobre animales como los de las fábulas de Esopo

Cuadro 2.9: Resumen del proyecto *Tale-Spin*

Debido al contexto histórico en que fueron desarrollados muchos sistemas usan lenguajes de marcos o redes semánticas para dar soporte a la representación de conocimiento.

Finalmente es necesario considerar también el tamaño y la diversidad de las bases de conocimiento utilizadas en cada proyecto. *Brutus* destaca por el manejo de mucha información sobre literatura, en concreto un vasto conocimiento lingüístico que hace uso de motivos literarios inspirados en argumentos clásicos, como las historias de traición entre amigos o parientes.

MakeBelieve utiliza una base de conocimiento con 9000 *reglas de causalidad*, tomadas de la base de conocimiento *ConceptNet*. Estas reglas establecen una relación causa-efecto, que el usuario-editor de la base de conocimiento introduce de la forma: $A \rightarrow_n B$ donde n es el número promedio de veces que suele emplearse esta implicación en una historia. Un ejemplo de estas implicaciones de sentido común es: "Una consecuencia de comer en un restaurante de comida rápida puede ser el estreñimiento". *ConceptNet* [LSE] es una base

de conocimiento de sentido común que adquiere la información a través de una comunidad web de instructores, por lo que está diseñada para ser muy escalable. Es un recurso libre pensado para utilizarse en sistemas de razonamiento de sentido común y procesamiento del lenguaje natural. Contiene un subconjunto de más de 400.000 sentencias, formando una red semántica de conocimiento muy extensa de libre disposición, con 280.000 proposiciones y una serie de herramientas para realizar inferencias sobre este conocimiento.

Materiales de procesamiento

Tras considerar la representación de conocimiento, el siguiente paso consiste en estudiar los mecanismos utilizados en su procesamiento, ya que ambas cuestiones están íntimamente relacionadas: *"Representation and reasoning are inextricably intertwined: we cannot talk about one without also, unavoidably, discussing the other"* [DSS93].

Los modelos de procesamiento resultan ser muy variados, pasando por el uso de sistemas de reglas de producción o razonadores lógicos a mecanismos más simples como los de la programación orientada a objetos, para proyectos no relacionados con la IA. David E. Rumelhart [Rum75] propuso un sistema tradicional de generación de historias basado en gramáticas de producción, que en el ámbito actual de este estudio se conocen como *gramáticas de historias*. Este enfoque es habitual en Lingüística Computacional aunque hoy día no esté muy extendido entre los sistemas actuales por ser poco flexible para generar historias variables y por tanto novedosas. Lee [Lee94] propone otro modelo narrativo basado en gramáticas de historias. *Joseph* [Lan97] es otro ejemplo de sistema clásico que utiliza una gramática de historias.

Debido al contexto histórico en que fueron desarrollados muchos sistemas usan *Lisp* (en algunos casos Prolog) para dar soporte al procesamiento del conocimiento. Actualmente los lenguajes más utilizados son *Java* y *C++*, aunque hay excepciones notorias, como *Mexica*, implementado íntegramente en *Pascal*.

Métodos de generación

Según Liu [LS02], los enfoques para generar historias pueden clasificarse en dos grupos: *estructuralista* y *transformacionalista*.

El estructuralismo usa algoritmos basados en la estructura de los dominios de conocimiento a los que pertenecen las historias, apoyados comúnmente en las las gramáticas de producción. El estructuralismo tiende a ser aplicado en los dominios más abstractos, como el puramente narrativo, tomando como punto de vista para la generación los objetivos globales que persigue el autor,

quien es también el encargado de asegurarse de que la estructura final generada sea apropiada. Habitualmente la estructura tiene forma de grafo, con un árbol principal que se construye de arriba a abajo aunque también sea posible construirlo al revés, de abajo a arriba. El ejemplo clásico de sistema estructuralista es *Automatic Novel Writer*.

El transformacionalismo usa algoritmos basados en realizar transformaciones sobre el estado de la historia en algunos de los dominios de conocimiento involucrados. Estos algoritmos suelen expresarse en forma de reglas que suelen modelar la sucesión cronológica de los procesos que se producen en el mundo ficticio de la historia. Para asegurar que la secuencia generada es apropiada a menudo se utilizan técnicas de evaluación o de planificación para controlar el modo en el que se ejecutan las reglas; y teniendo en cuenta que en casi todos los modelos de mundo ficticio habrá entidades autónomas, en ocasiones se simulará el comportamiento de cada entidad por separado.

El ejemplo clásico de sistema transformacionalista es *Tale-Spin*, siendo la primera y más conocida aproximación al paradigma transformacional de la generación de historias que tiene en cuenta dominios de conocimiento narrativos. Surgió como un paso más en los trabajos del grupo de Roger Schank en el terreno de la comprensión de historias. El sistema simula mediante reglas predefinidas un pequeño entorno habitado por personajes que evoluciona mientras que estos tratan de perseguir sus propios objetivos. Los personajes son animales personificados como los que aparecen en las fábulas de Esopo. Al comienzo de cada ejecución se pueden definir las características de los personajes, su conocimiento y sus objetivos; y durante el proceso el usuario puede participar ayudando a los personajes a realizar sus tareas, ya que ellos por sí solos no son capaces de tomar decisiones creativas. En este sistema se empiezan a modelar las relaciones sociales y las emociones que son fundamentales para enriquecer y dar sentido a una historia. El resultado se mantiene coherente gracias a un planificador que es el que en realidad guía el proceso de generación. Sin embargo no se aplica ningún mecanismo que evalúe el valor de la historia.

En realidad, en cualquiera de los dos grupos se pueden adoptar dos puntos de vista diferentes. El primero consiste en tomar como referencia al *autor*, haciendo que todo gire en torno a los objetivos narrativos generales, normalmente construyendo la estructura narrativa en forma de árbol. El segundo es tomar como referencia a las entidades del mundo ficticio, normalmente a los agentes que actúan como *personajes* de la historia, desarrollando la historia de acuerdo a la cronología ficticia y en función de los objetivos individuales de dichos personajes. En muchos casos estos dos puntos de vista se combinan, dado que hay características de la historia que se controlan mejor de forma global, como la coherencia, y otras que surgen mejor si son generadas

de forma local, como la credibilidad de un personaje.

Por otro lado, sistemas antiguos como *Roald* [Yaz89] ya propusieron combinar los paradigmas estructuralista y transformacionalista, generando historias resolviendo los conflictos que surgen entre los planes de los personajes, el plan del autor y una solicitud, o incluso todo un argumento, descrito inicialmente por el usuario.

Esto pone de manifiesto las dos posibilidades que se encuentran a la hora de crear algo nuevo, construirlo desde cero o modificar artefactos similares hasta obtener algo distinto. Los sistemas que modifican historias también resultan interesantes porque aprovechan el esfuerzo invertido en crear una historia de calidad y generan nuevas historias sin necesidad de conocer exactamente como se crean desde cero.

Con respecto a los algoritmos para generar historias existen diversos criterios para clasificarlos [LÖ4].

Lu y Singh [LS02] proponen una opción híbrida entre los enfoques estructuralista y transformacionalista, para aprovechar las ventajas de ambas filosofías. Utilizan el sistema *ConceptNet* para generar historias, aprovechando como se dijo anteriormente, un subconjunto de reglas de sentido común. Los autores consideran que las bases de conocimiento con información de sentido común son fuentes muy interesantes para obtener material con el que generar historias.

La primera distinción sería entre algoritmos con secuencias *explícitas* o *implícitas*. En los primeros se cuenta desde el principio con una estructura argumental en forma de árbol donde ya están representadas explícitamente todas las posibles combinaciones o secuencias de sucesos que pueden llegar a generarse. Por el contrario en los algoritmos de historia implícita, mucho más interesantes, la historia se construye a base de combinar elementos de acuerdo a el funcionamiento de un cierto algoritmo más o menos complejo.

Los sistemas más complejos combinan habitualmente diferentes etapas o fases, cada una con un algoritmo diferente, para generar y evaluar las historias generadas. *Brutus* o *Mexica* [PyPS01] son ejemplos de sistemas que no usan un único algoritmo para generar la historia completa.

Storybook [CL02] posee una arquitectura detallada para conseguir textos de calidad literaria en todos los estratos lingüísticos que conlleva la generación de historias. El generador de prosa narrativa se denomina *Author* y trata de combinar el enfoque descendente de las gramáticas productivas con el agrupamiento ascendente característico de los sistemas de generación de lenguaje natural. Trabajando sobre dominios muy limitados parece concentrar más esfuerzo en la generación de lenguaje natural que en la generación de la historia en sí.

Joseph también resulta interesante porque tiene una arquitectura dividi-

da en algo similar a componentes software: aparte de la gramática hay un modelo del mundo, un módulo de predicados temporales y el correspondiente intérprete de las órdenes del usuario.

Universe [Leb87], descrito en el Cuadro 2.10, es un generador de historias contemporáneo a *Tale-Spin*, aunque no ha llegado a ser tan divulgado como el primero.

Objetivos: Desarrollar una aplicación de narración automática **Material y método:** Sistema de reglas de producción implementado en Lisp que ejecuta el plan del autor sobre un mundo inicial con personajes y escenarios **Resultados:** Aplicación que recibe una orden Lisp por consola y devuelve el argumento para los capítulos de una telenovela en forma de oraciones simples en inglés

Cuadro 2.10: Resumen del proyecto *Universe*

La teoría narrativa que hay detrás de *Universe* es simple. Se basa en la idea de que los autores de las telenovelas añaden conflictos entre personajes y luego elaboran el argumento detallando la forma en que los personajes tratan de solucionarlos, añadiendo más conflictos según se resuelven los anteriores, estableciendo un ciclo de duración indefinida que permite generar todos los capítulos que sean necesarios.

Universe funciona en dos etapas. En la primera etapa define una red de personajes, representando las características y relaciones de cada uno mediante marcos semánticos. El punto fuerte del sistema es precisamente elaborar mucha información sobre cuales son los objetivos y las restricciones de cada uno de estos personajes en lo relativo al desarrollo de la historia. En la segunda etapa se utiliza un mecanismo de transformación del estado de dicha red de personajes para construir la historia. Las transformaciones obedecen a una serie de acciones ejecutadas por el autor computacional, quien persigue unos objetivos propuestos inicialmente por el usuario de acuerdo a lo que dicta un planificador interno.

Brutus es un sistema moderno de narración automática que trata de generar una historia creativa en relatos breves de calidad literaria comparable a la de un autor humano. *Brutus* separa la generación de la historia de la generación del discurso, donde aplica un claro enfoque estructuralista. En este sistema no existe simulación de ningún tipo de las entidades que pueblan el mundo ficticio.

Este sistema pretende conseguir más variabilidad que sus antecesores en las dimensiones de la historia y también incorpora un nuevo concepto de

diferenciación arquitectónica, lo que significa que durante el proceso de generación si un aspecto de la historia varía, el correspondiente componente de la arquitectura también debe variar.

Existe un argumento para la historia y otro elemento, que los autores llaman *story outline*, que se crea a base de seleccionar estructuras narrativas y expandirlas hasta que se consigue un plan para el discurso. Finalmente hay una etapa de generación de lenguaje donde la historia se combina con el discurso para obtener el resultado final.

Turner utiliza un enfoque KI-CBR en el modelo creativo del sistema *Minstrel*. Este modelo trata de hacer más original la historia y evitar que se generen soluciones triviales para los conflictos, tratando de ofrecer alternativas distintas a las conocidas por el sistema. El sistema funciona así: recibe una sentencia en lenguaje natural que pretende ser la moraleja de la historia que hay que contar. Luego se pone en marcha la parte creativa del programa para inventar la historia y después se "cuenta", lo cual es el resultado de resolver un problema de planificación y generar el texto en lenguaje natural. Las historias generadas por *Minstrel* persiguen cuatro objetivos de autor. En primer lugar las historias deben tratar un *tema*, esto es, deben contener un mensaje en forma de moraleja que se intenta transmitir al público. En segundo lugar las historias deben tener aspectos *dramáticos*, esto es, representar de forma literaria fenómenos como el suspense o la tragedia. En tercer lugar las historias deben ser *consistentes*, entendiendo esto como una coherencia en la lógica interna del mundo ficticio. En cuarto y último lugar la *presentación* de las historias, en este caso su lectura, debe resultar amena y agradable para el público.

Minstrel, para tratar de solucionar el problema de continuar el desarrollo de una historia de manera creativa, lo primero que hace es trasladar la situación actual a otro dominio. Este proceso de transformación del problema es la operación más delicada, ya que debe conducirnos a un dominio bien conocido donde se pueda encontrar fácilmente una solución para el problema transformado. Una vez hecho esto, se busca un caso similar y se recupera su solución, para después adaptarla al dominio original del problema. Esta adaptación resulta sencilla ya que consiste únicamente en hacer la traducción inversa desde el domino del problema transformado al dominio del problema original. Finalmente, si la solución resulta novedosa para el sistema, se guarda como un caso más en la base de casos.

La arquitectura de *Minstrel* está compuesta por esta serie de módulos: control, representación, memoria episódica, memoria semántica, TRAMs, planes del autor, lenguaje y un último módulo dedicado a otras utilidades.

La estructura de control se divide en tres niveles. El primero es el nivel de la *narración*. En este nivel, el más superficial, *Minstrel* actúa como un plani-

ficador: en cada ciclo elige el objetivo más prioritario de su agenda, encuentra los planes adecuados para lograr dicho objetivo y los va aplicando uno a uno hasta conseguir el éxito. Otra función de este nivel es la de fusionar el episodio que se recuperará de la memoria con el problema original. El segundo es el nivel de la *creatividad*. La parte creativa reside en la *memoria imaginativa*. Los planes del autor envían la especificación de un recuerdo a esa memoria, la cual intenta buscar la información en la memoria episódica. Si falla, se aplica una transformación TRAM a la especificación del recuerdo y se vuelve a consultar la memoria episódica. Estos pasos se repiten hasta que las combinaciones de TRAMs han creado un recuerdo que satisface la especificación o hasta que la profundidad de recursión supera un cierto límite. El tercero es el nivel de la *memoria episódica*. Este es el último nivel de control, la base de casos. Esta memoria es llamada desde la memoria imaginativa y también cada vez que se aplica un TRAM. Lo que hace es sencillamente *recuperación de casos*: buscar en un árbol algún episodio que coincida con la especificación del recuerdo que nos proponen.

Los métodos de transformación, recuerdo y adaptación (TRAMs, del inglés *Transform-Recall-Adapt Methods*) son la clave del modelo de creatividad de *Minstrel*. Se implementan como fragmentos de *Lisp* compilados y almacenados en colas.

Un TRAM generalmente se compone de un nombre, unas opciones para definir su comportamiento, una expresión para modificar el problema y otra para adaptar la solución encontrada.

Existe un caso base para los TRAMs que es el trivial, donde no es necesario modificar el problema ni adaptar la solución. El siguiente ejemplo puede servir para ilustrar la estructura de un TRAM.

```
(tram: rule standard-problem-solving (option :never-recurse)
(comment "Standard problem-solving recalls" "past episodes that
are exactly similar" "to the current episode, and which can be"
"used without any especial adaptation.") (transform *rspec*)
(adapt *episode*))
```

Existen otros TRAM más complicados, cuya función básicamente es la de trasladar el problema a otro dominio diferente mediante operaciones en *Lisp*, generalizándolo a base de eliminar restricciones y adaptando la solución otra vez al dominio original.

Para terminar se muestra un TRAM más complejo, que transforma un recuerdo sobre una acción no deliberada a uno donde la acción sí es deliberada.

```
(tram: rule intention-switch
(comment "To create a scene with an"
"intentional outcome, look for a scene"
"with the appropiate unintentional outcome"
"and adjust it accordingly.")
(option :dont-repeat)
;; This TRAM applies to acts with intended outcomes
(test (and (act? *spec*)
(inst:link *spec* &intends)))
;; Modify the *rspec* by switching the intended act
;; to a unintended one.
(recall
(let ((st (car (inst:link *rspec* &intends))))
(inst:delete-link *rspec* &intends st)
(inst:add-link *rspec* &unintended st)
*rspec*)
;; Adapt the recalled episodes by flipping the
;; intention back.
(adapt
(let ((st (car (inst:link *episode* &unintended))))
(inst:delete-link *episode* &unintended st)
(inst:add-link *episode* &intends st)
*episode*)))
```

Mexica [PyPS01] presenta un nuevo modelo de generación de historias algo más abstracto, que pretende simular el comportamiento de la creatividad humana, inspirándose en la forma en que los escritores construyen sus relatos. Este sistema genera esqueletos de relatos breves en lenguaje natural, utilizando un modelo cognitivo de escritura basado en ciclos iterativos que alternan una etapa de actividad y otra de reflexión (*engagement-reflection*).

Durante la *etapa de actividad*, *Mexica* genera material guiándose por una serie de restricciones retóricas y de contenido, evitando el uso explícito de objetivos o información sobre la estructura de la historia. A continuación se aplica un enfoque transformacionalista que utiliza un modelo emocional de los personajes y sus relaciones.

Durante la *etapa de reflexión* el sistema adopta el punto de vista del autor y se dedica a deshacer los bloqueos en el desarrollo de la historia que hayan podido producirse, evalúa la novedad y el valor de la historia que se está generando y verifica que los requisitos de coherencia se cumplen.

De esta forma, *Mexica* complementa y extiende los sistemas que se basaban en técnicas de resolución de problemas con objetivos explícitos que

conducían la generación de historias.

Mexica puede ejecutarse de acuerdo a cuatro modos de operación diferente. Los dos primeros sólo aprovechan la etapa de actividad. El primero se llama E1, que no usa filtros ni guías para la generación de la historia, y el segundo E2, que sí usa filtros y guías en forma de valores predefinidos que no cambian durante toda la generación. Los dos siguientes ya utilizan ambas etapas, de actividad y reflexión. El tercero se llama ER1 y no usa filtros ni guías para la generación de la historia, y el segundo ER2, que sí usa filtros y guías que se actualizan en cada etapa de reflexión.

Terminal Time [DMV02] es una aplicación de narración automática que combina eventos históricos y retórica ideológica para generar documentales televisivos de forma interactiva. La aplicación va narrando su historia y se detiene periódicamente en determinados puntos dejando que el público conteste a unas preguntas cuyas respuestas se tienen en cuenta para adaptar el resto de la historia. Esto hace que el resultado final sea diferente en cada ejecución y esté adaptado a la ideología dominante en el público. El sistema comienza recibiendo la solicitud inicial del público, en forma de preguntas sobre el género, raza, tecnología, clase y religión del usuario, además de una indicación sobre cómo debe ser el arco narrativo (si debe progresar o declinar). En una segunda interacción, el tema ideológico se refina mediante otras preguntas, pudiendo introducir un subtema, combinando algunos aspectos del usuario Finalmente durante la tercera interacción continúa refinando el tema ideológico e introduce la posibilidad de cambiar el arco narrativo.

Métodos de evaluación

Muchos proyectos de narración automática carecen de un modelo de evaluación como parte de su metodología de desarrollo. Como es habitual con los sistemas creativos, los productos resultan más accesibles que el propio proceso, por lo que para evaluar una aplicación de narración automática, al menos desde el punto de vista del usuario, se hace necesario evaluar las historias generadas.

Prácticamente no existen proyectos de investigación que evalúen rigurosamente el proceso de generación de historias de una aplicación de narración automática. A lo sumo se proporcionan datos básicos, como los requisitos hardware o la duración del proceso, lo que ocurre en la documentación técnica de *Minstrel* [Tur92]. Las excepciones se encuentran en herramientas orientadas al autor humano, como es el caso de *Wide Ruled* [SJMM07], donde se evalúa de forma indirecta la capacidad de uso y aprendizaje que tiene el modelo interno basado en las transformaciones de estado de los personajes, los escenarios y el propio argumento.

Minstrel y *Mexica* integran mecanismos para evaluar la historia según se desarrolla, y se señalan al menos algunos rasgos que permiten reconocer el valor de una historia. Aunque muchos, como el significado profundo de una historia o las emociones que evoca en el público, no son modelados internamente al sistema, hay otros que sí lo son como los que afectan al discurso y a la presentación textual de las historias.

Minstrel propone un método de evaluación basado en la valoración que varias personas tengan sobre las historias generadas. Nueve personas respondieron a un cuestionario sin saber que las historias habían sido generadas por ordenador. Aparte de responder preguntas sobre la edad, la educación y el sexto del hipotético autor de la historia, también se preguntaba acerca de la puntuación general que merecía la historia, la inteligencia del argumento, la atención que se presta a los detalles, la coherencia y el uso del lenguaje.

Mexica propone un método de evaluación similar al de *Minstrel*, cambiando las preguntas del cuestionario, para que se valore el flujo narrativo, la estructura la calidad del contenido, el suspense y finalmente la calidad final de la historia.

Aún menos considerado que el valor de una historia es su novedad. Prácticamente todos los sistemas, excepto aquellos que generan versiones de la misma fábula, tratan de generar fábulas distintas en cada ejecución, aunque confían simplemente en los procesos combinatorios implementados para asegurarla. Pocos proyectos estudian como generar discursos distintos para cada fábula generada y menos aún son los que pretenden ser creativos en la forma de escoger la presentación final para un discurso dado.

Una última cuestión relacionada con la evaluación es el aprendizaje automático. Esta caracteristica resulta posible cuando existe un mecanismo de autoevaluación integrado en la aplicación, como en el caso de *Minstrel*. Este sistema es capaz de aprender de sus propias creaciones añadiendo casos a su base de casos en la última etapa del ciclo CBR.

2.2.4. Resultados

En este apartado se muestran los resultados de los proyectos de investigación analizados. En la mayor parte de los casos se tratan de aplicaciones de narración automática que generan historias textuales, aunque también existen proyectos que trabajan con otras tecnologías de presentación, como imágenes, videos o incluso animaciones 3D generadas también de forma automática.

En el Cuadro 2.11 se presenta un fragmento de la salida en texto plano que genera *Automatic Novel Writer*. Como resulta evidente el sistema genera una secuencia de eventos expresados con frases muy cortas en un inglés muy

pobre, sin uso de técnicas demasiado avanzadas de generación de lenguaje para enriquecer el resultado.

> The day was Monday. The pleasant weather was sunny. Lady Buxley was in a park. James ran into Lady Buxley. James talked with Lady Buxley. Lady Buxley flirted with James. James invited Lady Buxley. James liked Lady Buxley.Lady Buxley liked James. Lady Buxley was with James in a hotel. Lady Buxley was near James. James caressed Lady Buxley with passion. James was Lady Buxley's lover. Marion following them saw the affair. Marion was jealous.

Cuadro 2.11: Historia generada mediante *Automatic Novel Writer*

Los resultados de *Tale-Spin* son coherentes aunque poco atractivos, como puede comprobarse en el Cuadro 2.12. Con respecto a la presentación, se aprecia el uso de plantillas para embellecer algunas frases, como ocurre con el clásico comienzo "Erase una vez...".

> Once upon a time George Ant lived near a patch of ground. There was a nest in an ash tree. Wilma Bird lived in the tree. There was some water in the river. Wilma knew that the water was in the river. George knew that the water was in the river. One day Wilma was very thirsty. Wilma wanted to get near some water. Wilma flew from her nest across a meadow through a valley to the river. Wilma drank the water. Wilma wasn't very thirsty any more.

Cuadro 2.12: Historia generada mediante *Tale-Spin* [Mee81]

Algo similar ocurre con las historias generadas por *Joseph*, como puede verse en el ejemplo del Cuadro 2.13.

En el Cuadro 2.14 se recoge un ejemplo de historia generada por *Brutus*, sin duda una de las más convincentes en cuanto a presentación lingüística. Los autores del sistemas llegaron incluso a proponer una competición contra escritores humanos.

Storybook [CL02] genera varias historias que son siempre versiones de la misma fábula: *La Caperucita Roja*. En el Cuadro 2.15 se proporciona un ejemplo que ilustra el buen manejo de las presentación lingüística.

once upon a time there lived a peasant was married to wife. wife always disobeyed peasant. one day it happened that peasant quarreled with wife. when this happened peasant felt distress. in response peasant took a walk in the woods. peasant found a pit when he looked under the bush. when this happened, peasant desired to punish wife. in response, peasant made it his goal that wife would be in the pit. peasant tricked wife. wife was in the pit. peasant was alone.

Cuadro 2.13: Historia generada mediante *Joseph* [Lan97]

Universe genera argumentos para los capítulos de una interminable telenovela. La salida del sistema combina partes en lenguaje natural con otra información más técnica sobre los objetivos y los planes que se ejecutan internamente, como puede verse en el Cuadro 2.16. La primera línea corresponde a la orden *Lisp* a la que está respondiendo el sistema.

En el Cuadro 2.17 se muestra un ejemplo de historia generada con *Minstrel*. Las historias versan sobre las leyendas del rey Arturo y los Caballeros de la Mesa Redonda. El título "La princesa vengativa" y la moraleja "El engaño es un arma difícil de usar" son frases que en ocasiones se añaden manualmente para adornar esta salida. Los resultados de evaluación de este sistema sugieren que las historias son bastante coherentes, aunque el uso del lenguaje es poco acertado y la historia en general no es muy bien valorada (1.5 sobre 5). La historia es comprensible y tiene un claro hilo conductor que transmite adecuadamente el mensaje de la moraleja.

Minstrel se implementó en Common Lisp, mediante 8000 líneas de código útiles más un paquete de utilidades para la representación de conocimiento llamado *Rhapsody*. Para generar una historia corta como la del Cuadro 2.17 Minstrel tardaba cerca de los 30 minutos en una estación de trabajo Sun 3/50 de 1'5 MIPS. Muchos proyectos no aportan información sobre los resultados en cuestión de eficiencia de sus aplicaciones de narración automática, aunque otros demuestran que son capaces de trabajar en tiempo real, como es el caso de muchos proyectos de narración interactiva dirigida automáticamente.

En el Cuadro 2.18 se muestra un fragmento de historia generada por *Mexica*. En el modo de ejecución ER2 *Mexica* obtiene, al ser evaluado, valores comparables e incluso ligeramente superiores, en el caso de flujo narrativo, contenido y calidad final, a los obtenidos por un autor humano que edita manualmente una historia similar. Pérez y Pérez [PyPS04] usa el mismo método para evaluar también las historias generadas por Minstrel y los valores

Dave Striver loved the university - its ivy-covered clocktowers, its ancient and sturdy brick, and its sun-splashed verdant greens and eager youth. The university, contrary to popular opinion, is far from free of the stark unforgiving trials of the business world: academia has its own tests, and some are as merciless as any in the marketplace. A prime example is the dissertation defense: to earn the PhD, to become a doctor, one must pass an oral examination on one's dissertation. This was a test Professor Edward Hart enjoyed giving.

Dave wanted to be a doctor. But he needed the signatures of three people on the first page of his dissertation, the priceless inscriptions which, together, would certify that he had passed his defense. One of the signatures had to come from Professor Hart, and Hart had often said - to others and to himself - that he was honored to help Dave secure his well-earned dream.

Well before the defense, Striver gave Hart a penultimate copy of his thesis. Hart read it and told Dave that it was absolutely first-rate, and that he would gladly sign it at the defense. They even shook hands in Hart's book-lined office. Dave noticed that Hart's eyes were bright and trustful, and his bearing paternal.

At the defense, Dave eloquently summarized Chapter 3 of his dissertation. There were two questions, one from Professor Rodman and one from Dr. Teer; Dave answered both, apparently to everyone's satisfaction. There were no further objections.

Professor Rodman signed. He slid the tome to Teer; she too signed, and then slid it in front of Hart. Hart didn't move.

"Ed?" Rodman said.

Hart still sat motionless. Dave looked at him.

"Edward, are you going to sign?"

Later, Hart sat alone in his office, in his big leather chair. He tried to think of ways he could help Dave achieve his goal.

Cuadro 2.14: Historia generada mediante *Brutus* [BF99]

Once upon a time, a woodcutter and his wife lived in a small cottage. The woodcutter and his wife had a young daughter, whom everyone called little Red Riding Hood. She was a merry little maid, and all day long she went singing about the house. Her mother loved her very much.

Cuadro 2.15: Historia generada mediante *Storybook* [CL02]

(tell '((churn JOSHUA FRAN)) (together JOSHUA and VALERIE))))
working on goal—DO-DISABLE FRAN
FRAN has a spinal injury and is paralysed
FRAN doesn't want to ruin J'life
FRAN pretends to blame JOSHUA for her malady
working on goal –DUMP-LOVER FRAN JOSHUA
using plan BREAK-UP DUMPER/FRAN DUMPED/JOSHUA
FRAN tells JOSHUA she doesn't love him any more

Cuadro 2.16: Historia generada mediante *Universe* [Leb87]

Once upon a time there was a Lady of the Court named Jennifer. Jennifer loved a knight named Grunfeld. Grunfled loved Jennifer. Jennifer wanted revenge on a lady of the court named Darlene because she had the berries which she picked in the woods and Jennifer wanted to have the berries. Jennifer wnated to scare Darlene. Jennifer wanted a dragon to move towards Darlene so that Darlene believed it would eat her. Jennifer wanted to appear to be a dragon so that a dragon would move towards Darlene. Jennifer drank a magic potion. Jennifer transformed into a dragon. A dragon moved towards Darlene. A dragon was near Darlene.
Grunfeld wanted to impress the king. Grunfled wanted to move towards the woods so that he could fight a dragon. Grunfeld moved towards the woods. Grunfeld was near the woods. Grunfeld fought a dragon. The dragon died. The dragon was Jennifer. Jennifer wanted to live. Jennifer tried to drink a magic potion but failed. Grunfled was filled with grief.
Jennifer was buried in the woods. Grunfeld became a hermit.

Cuadro 2.17: Historia generada mediante *Minstrel* [Tur92]

resultan inferiores a los de *Mexica* y del autor humano.

Existen sistemas que, debido a una orientación diferente en la forma en que presentan las historias al público, generan imágenes o animaciones en vez de texto. *ANI* utiliza figuras geométricas para representar a los personajes del cuento *Blancanieves y los siete enanitos*. El sistema recibe por parte del usuario el argumento descrito a grandes rasgos de una versión simplificada de dicha cuento, y produce la animación necesaria para narrarlo de manera visual. *IDIC* [SD94] está inspirado en *Tale-Spin*, como tantos otros, pero sin embargo no genera lenguaje natural sino imágenes. Al no crearse ninguna imagen nueva, lo que se consigue con este sistema es crear un nueva historia a base de juntar fragmentos de otras historias.

El sistema *ISRST* [NI07], descrito en el Cuadro 2.19, presenta las historias de forma multimodal, con videos formados por transiciones entre imágenes estáticas y texto superpuesto, como puede verse en la Figura 2.1.

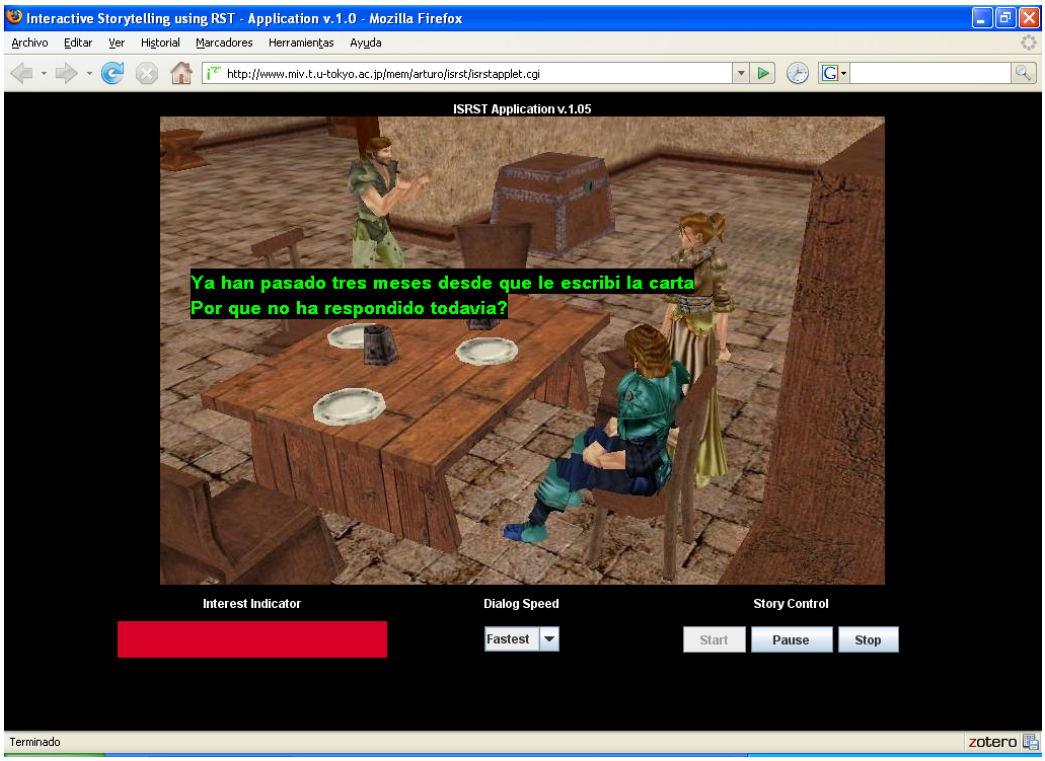

Figura 2.1: La aplicación *ISRST* narrando una animación

Otro ejemplo de aplicación que genera historias en forma de animación visual es *SWAN* [LZ02], como se ilustra en la Figura 2.2.

Jaguar_knight was an inhabitant of the great Tenochtitlan. Princess was an inhabitant of the great Tenochtitlan. From the first day they met, Princess felt a special affection for Jaguar_knight. Although at the beginning Princess did not want to admit it, Princess fell in love with Jaguar_knight. Princess respected and admired Artist because Artist's heroic and intrepid behaviour during the last Flowery-war. For long time Jaguar_knight and Princess had been flirting. Now, openly they accepted the mutual attraction they felt for each other.
Jaguar_knight was an ambitious person and wanted to be rich and powerful. So, Jaguar_knight kidnapped Artist and went to Chapultepec forest. Jaguar_knight's plan was to ask for an important amount of cacauatl (cacao beans) and quetzalli (quetzal) feathers to liberate Artist. Princess had ambivalent thoughts towards Jaguar_knight. On one hand princess had strong feelings towards Jaguar_knight but on the other hand Princess abominated what Jaguar_knight did.
Suddenly, the day turned into night and after seconds the sun shone again. Princess was scared. The Shaman explained to Princess that Tonatiuh (the divinity representing the sun) was demanding Princess to rescue Artist and punish the criminal. Otherwise Princess's family would die.
Early in the Morning Princess went to Chapultepec forest. Princess thoroughly observed Jaguar_knight. Then, Princess took a dagger, jumped towards Jaguar_knight and attacked Jaguar_knight. Jaguar_knight was shocked by Princess's actions and for some seconds Jaguar_knight did not know what to do. Suddenly, Princess and Jaguar_knight were involved in a violent fight. In a fast movement, Jaguar_knight wounded Princess. An intense haemorrhage arose which weakened Princess. Jaguar_knight felt panic and ran away.
Thus, while Tlahuizcalpantecuhtli (the god who affected people's fate with his lance) observed, Princess cut the rope which bound Artist. Finally, Artist was free again! Princess was emotionally affected and was not sure if what Princess did was right. Princess was really confused. The injuries that Princess received were very serious. So, while praying to Mictlantecuhtli (the lord of the land of the dead) Princess died.

Cuadro 2.18: Historia generada mediante *Mexica* [PyPS01]

Objetivos: Narración interactiva realimentada por el público **Material y método:** Teoría de la estructura retórica del discurso, razonamiento sobre RDF **Resultados:** Drama de situación llamado *When your Heart takes over* y documental divulgativo llamado *The Scientific Case for Creation*

Cuadro 2.19: Resumen del proyecto *ISRST*

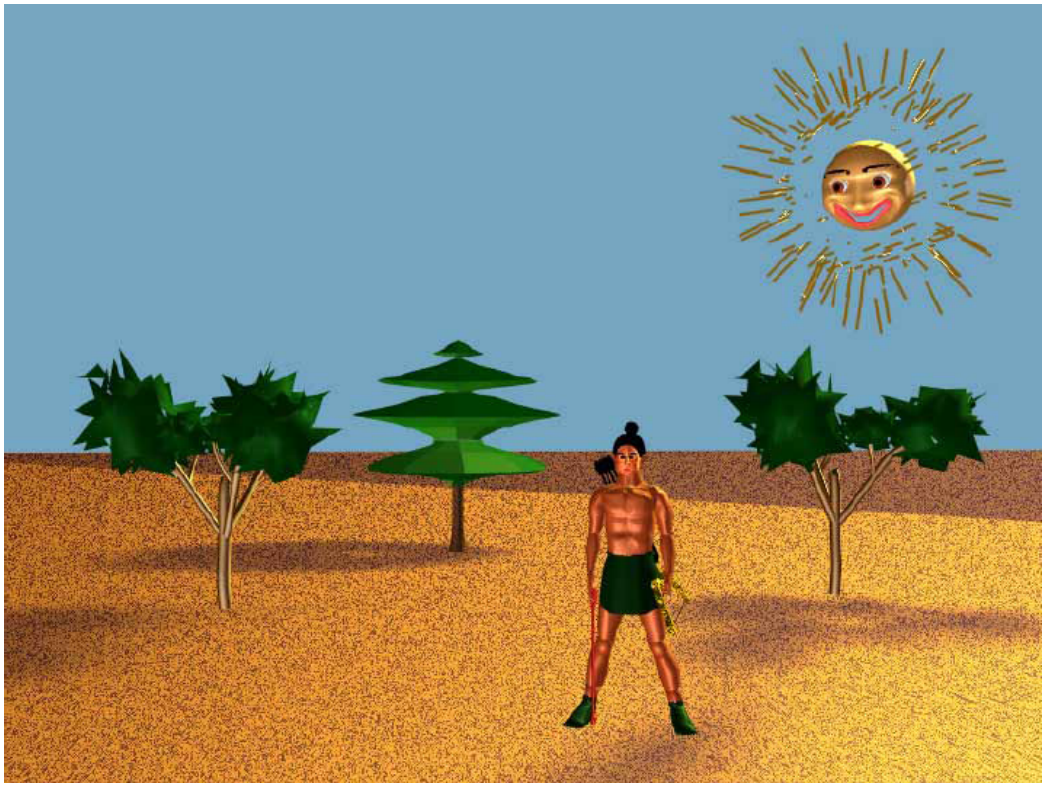

Figura 2.2: Animación generada mediante el sistema *SWAN*

2.2.5. Estado de la cuestión

Hoy día la mayoría de los proyectos asumen que una aplicación de narración automática debe responder al menos a una solicitud inicial del usuario que debe afectar de forma significativa a las posibles historias generadas. En el sistema que propone *Lee*, por ejemplo, se parte del esqueleto del argumento que proporciona el usuario, para después crear un mundo para los personajes donde finalmente se desarrollen sus interacciones. Este tipo de solicitud inicial es la más habitual en los proyectos que no ofrecen interacción durante el desarrollo de la historia.

En otros casos la participación del público puede ir más lejos, y sin llegar a ofrecer verdadera narración interactiva e inmersiva, el usuario puede intervenir en varias ocasiones durante el desarrollo de la historia, tomando decisiones sobre cómo ha de continuar. A menudo esto no es más que dejar en manos del usuario una decisión que de otra forma sería tomada por el sistema, bien aleatoriamente, bien usando el mecanismo habitual con el que se toman las demás decisiones.Por ejemplo, en *IDIC* el usuario actúa como el director de una película durante el montaje, seleccionando y ordenando las imágenes que compondrán la historia. El sistema *Joseph* genera de forma interactiva cuentos populares rusos, permitiendo al usuario intervenir en el proceso en forma de elecciones en la fábula, aunque no le permita realizar modificaciones en el discurso. Otros sistemas, como *IST* [GBPMRI07] o *Wide Ruled* [SJMM07] también hacen lo mismo: parten de un modelo de generación secuencial y lo modifican para que ciertas decisiones sean tomadas por el usuario de forma interactiva.

2.3. Descripción de las herramientas

En este apartado se describen brevemente aquellas herramientas, ya sean teóricas como las lógicas descriptivas, o prácticas como el razonador Pellet, que se mencionan en los apartados 2.2 y 3.2 y cuyo conocimiento es crucial para poder comprender el material y el método de investigación de este trabajo, así como la discusión correspondiente del capítulo 5.

En cada apartado primero se describen los modelos y formalismos teóricos a los que se hace referencia en este trabajo, seguidos de los lenguajes y los sistemas software que los implementan. Este material se describe sólo con el nivel de detalle suficiente para entender cual es la función que desempeñan en esta investigación.

2.3.1. Ontologías y bases de conocimiento

Los sistemas basados en conocimiento tienen que resolver dos problemas fundamentales: la representación del conocimiento y su procesamiento para utilizarlo en la resolución de una tarea concreta. Estos sistemas tienen que gestionar su base de conocimiento (KB) de una forma coherente, organizando su contenido según algún criterio razonable. Uno de esos posibles criterios es el uso de ontologías, un modelo de conceptualización que se explica a continuación.

En Filosofía, Ontología significa "el estudio del ser", una rama de la Metafísica general que se ocupa en definir lo que existe y en establecer categorías fundamentales para todas las cosas en función de sus propiedades. En Inteligencia Artificial, Ontología hace referencia a una representación explícita y expresada en un lenguaje formal del modelo conceptual de un cierto dominio de conocimiento. Gruber da la siguiente definición: *"A formal, explicit specification of a shared conceptualization"* [Gru93]. Este carácter de compartición es lo que permite reutilizar el vocabulario semántico de una ontología en la construcción de diferentes bases de conocimiento e incluso de diferentes sistemas inteligentes, siempre que estos trabajen con la misma perspectiva sobre un mismo dominio de conocimiento.

Las ontologías no se limitan a clasificar categóricamente un conjunto de conceptos, sino que definen como son las relaciones y los axiomas lógicos que conectan unos con otros, generalmente en términos de los individuos de las bases de conocimiento que se añadirán después para formar el modelo completo y que podrán o no ser aceptados como instancias válidas de dicha conceptualización previa.

Las ontologías se pueden clasificar según los dominios de conocimiento que intentan abarcar o las aplicaciones para las que están diseñadas. Algunas de estas clasificaciones distinguen casos particulares como las *ontologías lingüísticas*, las *ontologías de métodos de resolución de problemas* (PSMs, del inglés *Problem-Solving Methods*), que formalizan el vocabulario, independiente del dominio del problema, necesario para resolver dicho problema, las *ontologías generales*, que modelan conceptos abstractos de propósito general y subsumen las conceptualizaciones más específicas de otras ontologías, y las *metaontologías*, dedicadas a modelar el vocabulario necesario para describir otras ontologías de dominios concretos.

La disciplina que estudia el ciclo entero de desarrollo de las ontologías y las diversas metodologías para llevarlo a cabo se conoce como Ingeniería Ontológica, un tipo particular de Ingeniería del Conocimiento. En ella se estudia la manera de dividir las ontologías en módulos reutilizables y mantenibles.

Existen numerosas metodologías para el desarrollo de ontologías, las cua-

les siguen principios de diseño como los que se documentan en el proyecto *GOP* (*Game Ontology Project* [ZMFV+05]). Los principios de esta metodología, basada en la teoría de prototipado y clasificación de George Lakoff [Lak87], son seis. El primer principio es el de la *economía cognitiva*, según el cual cada concepto de la ontología debe ser lo suficientemente específico como para reflejar toda la información esencial pero a la vez lo suficientemente general como para no contener detalles irrelevantes. El segundo principio es el de *centralidad*, según el cual algunos individuos son mejores ejemplos (ejemplos más centrales) de un concepto que otros. El tercer principio es el de *grado de pertenencia*, según el cual los individuos tiene grado de pertenencia a un concepto y por lo tanto no existen fronteras nítidas entre conceptos. El cuarto principio es el de *grado de centralidad*, según el cual los individuos y subconceptos pueden ser más o menos centrales a un concepto. El quinto principio es el del *razonamiento metonímico* o de punto de referencia, según el cual los subconceptos pueden actuar como su correspondiente superconcepto en cierto tipo de procesos de razonamiento. Y finalmente el sexto principio es el de *categorización en niveles básicos*, según el cual los conceptos se organizan jerárquicamente, de manera que los conceptos cognitivos básicos queden en el medio de una jerarquía que va de más general a más específico.

La modularización de ontologías que propone Alan Rector [Rec03] es también un referente importante. Según este autor es conveniente *normalizar* una ontología para hacerla más usable y funcional. Los criterios para realizar esta normalización son cuatro. El primer criterio es partir de una taxonomía esquelética formada únicamente por conceptos primitivos, donde ninguno de ellos tiene más de un padre. El segundo criterio es hacer que cada rama se corresponda con una división lógica y homogénea del dominio. El tercer criterio es mantener dos ramas distintas de conceptos primitivos desde la raíz de la ontología: una de conceptos "autocontenidos", disjuntos pero sin cubrirse completamente unos a otros en particiones exhaustivas, y otra de conceptos de "refinamiento", tipos de valores (disjuntos o no) que tienen como hijos particiones disjuntas y exhaustivas. Finalmente el cuarto criterio es procurar que ningún axioma o restricción de dominio o rango permita inferir que un concepto primitivo tiene más de un padre.

Existen ontologías, habitualmente generales o sobre dominos muy extensos, de uso muy difundido entre los sistemas inteligentes. Algunos ejemplos de estas ontologías son *The Generalized Upper Model* (GUM) [BHR95], *Cyc* [Cyc] o *Mikrokosmos* [Nir]. También pueden encontrarse numerosos recursos sobre ontologías en Internet, siendo uno de los portales más veteranos el mantenido John Bateman [Bat].

A continuación se introducirán algunos conceptos básicos para representar sucesos de una forma computacional, seguidos de una descripción de las

herramientas disponibles para editar ontologías y bases de conocimiento como las que se mencionarán a lo largo de este trabajo.

Representación de sucesos

La representación de sucesos en el tiempo que afectan a entidades en el espacio es un problema común a muchos sistemas basados en conocimiento.

Aunque a menudo cada proyecto utiliza su propia representación para modelar los sucesos que pretende tratar, se ha considerado importante describir aquí una teoría clásica aunque muy popular de representación de conocimiento que servirá para ilustrar las cuestiones fundamentales sobre la representación de sucesos y ayudará a entender las decisiones que se toman en este, y otros trabajos de investigación, al respecto. Esta teoría se conoce como la Teoría de la Dependencia Conceptual.

Originalmente propuesta por Schank [Sch69] y su equipo para la formalización del conocimiento adquirido a través de la comprensión del lenguaje natural, esta teoría ha ido sufriendo actualizaciones [Lyt92], aunque en la práctica es poco utilizada debido a su excesiva complejidad. Sin embargo algunas de sus ideas fundamentales se encuentran en numerosos trabajos de investigación, no necesariamente relacionados con la comprensión o la generación del lenguaje natural.

La Teoría de la Dependencia Conceptual pretende ser un modelo independiente del lenguaje que sirva para expresar mediante diagramas el significado de cualquier frase en lenguaje natural. El modelo se estructura en forma de varios nodos de información interconectados entre sí y un conjunto finito de acciones primitivas con sus correspondientes modificadores. Las frases se descomponen previamente en los sucesos reales o figurados a los que hacen referencia. Dichos sucesos serán nodos o conjuntos de nodos del modelo que representarán acciones y se conectarán con los nodos correspondientes a los objetos y agentes que involucran. Además se añadirán atributos a dichas acciones y objetos, además de relaciones temporales y espaciales.

Los sucesos se representan con una o varias primitivas conectadas y con modificadores, una estructura más grande cuanto más complicada sea la semántica del suceso que se pretende describir. La lista de primitivas se presenta en el Cuadro 2.20.

Editores de ontologías y bases de conocimiento

Protégé [GMF+02, CDF+], es una herramienta de desarrollo de ontologías creada por el grupo *Stanford Medical Informatics*. Esta herramienta funciona internamente con una representación basada en marcos e implementada

Primitiva	Descripción
ATrans	Transferencia de una relación abstracta
PTrans	Transferencia de la localización física de un objeto
Propel	Aplicación de una fuerza física a un objeto
MTrans	Transferencia de información mental
MBuild	Construcción de nueva información mental
Speak	Pronunciación de un sonido
Attend	Enfoque de un sentido sobre un estímulo
Move	Movimiento de una parte del propio cuerpo
Grasp	Agarrar un objeto
Ingest	Ingerir un objeto
Expel	Deshacerse de un objeto del propio cuerpo

Cuadro 2.20: Lista de primitivas de la Teoría de la Dependencia Conceptual

en *CLIPS*. Lleva muchos años en funcionamiento y ha ido formando una comunidad de usuarios muy importante.

Se distribuye como una única aplicación *Java* de código libre que se ejecuta localmente en la máquina del cliente. Sin embargo se comporta como una herramienta híbrida al distribuirse con dos variantes, una que permite representar, cargar y guardar el conocimiento en RDF y otra más moderna llamada *Protégé-OWL*, que incluye un plug-in para poder hacer lo mismo en OWL. Permite editar y visualizar clases, propiedades, individuos y demás expresiones lógicas posibles mediante un interfaz gráfico muy agradable e intuitivo. Su arquitectura está basada en plug-ins, pensados para realizar funciones muy diferentes. Muchos están pensados para el marcado semántico de documentos, y existen otros que permiten editar y ejecutar reglas SWRL. Permite conectarse fácilmente con razonadores de lógicas descritivas que acepten la interfaz de comunicación DIG. *Protégé* proporciona una potente API llamada *Protégé-OWL API*, para trabajar con las tecnologías asociadas a los lenguajes RDF y OWL, cargar, salvar, consultar y realizar modificaciones en bases de conocimiento que sigan estos modelos, así como comunicarse con razonadores de lógicas descriptivas. Este API puede utilizarse para desarrollar plug-ins que se ejecuten como componentes integrados en la interfaz de usuario de *Protégé* o para desarrollar aplicaciones independientes.

En cuanto a visualización de ontologías y bases de conocimiento, *Protégé* cuenta con un plug-in llamado *Jambalaya* [CHI] que permite representar y editar gráficamente los ficheros que está creando el usuario. También ofrece la posibilidad de generar automáticamente documentación para ontologías y

bases de conocimiento en formato *OWLDoc* [CO].

SWOOP [Minb] es un editor de ontologías *OWL* alternativo, ligero y basado en conceptos de diseño y navegación hipermedia. Fue creado por el laboratorio *Mindswap* y su aspecto es similar a un navegador web, pudiendo recorrer los distintos elementos de las distintas ontologías cargadas a base de activar hipervínculos. Admite varias sintaxis de presentación, desde la sintaxis abstracta de OWL, a la sintaxis habitual RDF/XML, un formato de tripletas, etc. Incluye el razonador *Pellet* integrado, con la posibilidad de depurar la ontología al tiempo que se está creando.

Las limitaciones de DIG han hecho que editores como *Protégé* o *SWOOP* ofrezcan alternativas para conectar de una manera más directa con los razonadores. Para resolver este problema, se desarrolla una nueva versión del estándar (DIG 2.0) que supera todas esas limitaciones.

2.3.2. Lógicas descriptivas

Un sistema lógico o una *lógica* es un formalismo para representar conocimiento que se define mediante una determinada sintaxis, semántica y conjunto de reglas de inferencia. Estas reglas han de ser *correctas* y *completas* si se desea que los procesos de inferencia resulten siempre infalibles, como ocurre en muchos de los proyectos de IA.

Las lógicas descriptivas (DLs, del inglés *Description Logics* [BCM+03]) son una familia de lógicas que se obtienen al imponer ciertas restricciones a la lógica de primer orden.

Las DLs provienen de otros formalismos anteriores como las redes semánticas y los sistemas de marcos, a los que se les añade semántica formal. Son la base de lenguajes de representación de conocimiento muy potentes que se utilizan actualmente.

Según define la IA clásica, el conocimiento es un conjunto de datos con los que un sistema inteligente describe internamente la situación del mundo al que tiene acceso, manifestándose dicha inteligencia en forma de razonamiento lógico, inferencias, deducciones, etc., que el sistema es capaz de realizar de forma automática. Tradicionalmente dichos sistemas inteligentes se construían en base a formalismos sintácticos cuya semántica se proporcionaba expresamente para cada sistema. Hoy día las DLs permiten establecer una semántica estándar para estos formalismos.

En las DLs todos los predicados contienen una o dos variables, existiendo sólo tres tipos de elementos básicos: los conceptos, los roles y los individuos.

- Los *conceptos* son descripciones con una estructura potencialmente compleja formada a base de componer otras descripciones más sim-

ples con un conjunto limitado de operadores. En un sistema lógico el concepto y su definición constituyen un axioma terminológico.

- Los *roles* son términos formales que establecen las relaciones de los conceptos entre sí o con tipos primitivos.
- Los *individuos* son asertos o construcciones formales simples que representan directamente objetos del mundo, es decir, instancias de los conceptos.

Los conceptos se comportan como conjuntos que contienen individuos, de los cuales decimos que son instancias del "tipo" del concepto. Los conceptos pueden ser primitivos o definidos. Los conceptos primitivos, por norma general, sólo contienen aquellos individuos que han sido asertados de forma explícita como tales mientras que los conceptos definidos, además, pueden reconocer como instancias suyas a otros individuos de la base de conocimiento que cumplan ciertas restricciones universales, existenciales o de cardinalidad sobre el uso de sus roles. Esto se traduce en que, mientras que sólo se pueden *declarar* restricciones *necesarias* de consistencia para los individuos que pertenezcan a conceptos primitivos, los conceptos definidos precisamente reciben ese nombre porque se *definen* en función de condiciones *necesarias y suficientes* que determinan si un individuo de la base de conocimiento pasa o no pasa a ser considerado instancia de dicho concepto.

Los conceptos se organizan de acuerdo a una taxonomía polijerárquica, es decir, donde cada concepto puede tener uno o más padres. En esta taxonomía los conceptos más generales aparecen más cerca de la raíz y los más específicos cerca de las hojas.

Los individuos también pueden tener uno o más tipos pudiéndose encontrar, por lo tanto, relacionados con conceptos de cualquier punto de la taxonomía.

Los roles también se organizan de acuerdo a una taxonomía donde cada rol sólo puede tener un rol padre y del cual heredará algunas propiedades, como las restricciones de dominio y rango.

Teniendo en cuenta estas explicaciones, a menudo se distinguen tres repositorios dentro de un modelo basado en DLs llamados TBox, RBox y ABox.

- *TBox* es la caja terminológica, el esquema conceptual que permite hacernos una idea del tipo de individuos que vamos a poder reconocer en la base de conocimiento, según las propiedades lógicas que cumplan.
- *RBox* es la caja relacional, la taxonomía de roles que se utilizan para definir conceptos y declarar sus propiedades, además de para relacionar unos individuos con otros en la base de conocimiento.

- *ABox* es la caja asertiva, el conjunto de individuos de la base de conocimiento y sus relaciones.

Estos tres repositorios forman un modelo basado en DLs, que se considera consistente si todos los conceptos de TBox son satisfactibles y el contenido de ABox no infringe ninguna de las restricciones declaradas en TBox o en RBox. A menudo los sistemas basados en DLs se complementan con *atributos*, relaciones entre individuos y valores de todo tipo (booleanos, numéricos, textuales, etc.), que sirven para aumentar la expresividad de la representación aunque a menudo no se tengan en cuenta durante los procesos de razonamiento.

Los razonadores basados en DLs son algoritmos de implementación compleja altamente optimizada para que se puedan realizar de forma eficaz y automática la validación de la consistencia del modelo, la clasificación de la taxonomía conceptual y el reconocimiento de los individuos según sus tipos inferidos. La eficiencia de estos procesos de razonamiento es variable dependiendo de la complejidad del modelo.

El razonamiento con DLs tiene dos cualidades muy peculiares que afectan notablemente al diseño de sistemas inteligentes basados en estos formalismos:

- Se asume que el mundo es abierto, lo que significa, a diferencia de la premisa de mundo cerrado, que el desconocimiento de un dato no implica que este sea falso. El sistema está abierto permanentemente a que se aserten nuevos conceptos, roles e individuos relacionados en la base de conocimiento y por lo tanto el sistema no inferirá que algo es cierto o falso hasta que tenga datos que permitan demostrarlo.

- El razonamiento es monótono, lo que quiere decir que se asume que las verdades inferidas en un modelo seguirán siendo verdad aunque se aserte nueva información en la base de conocimiento.

Ambas cualidades otorgan las siguientes propiedades al sistema: por un lado si se introducen nuevos asertos en la base de conocimiento que entran en contradicción con las antiguas verdades del modelo y lo hacen inconsistente, este lo seguirá siendo por muchos más asertos que se añadan; pero por otro lado si se retractan asertos de un modelo consistente, siempre se llegará a otro modelo también consistente.

Este formalismo de la lógica clásica se ha hecho muy popular en nuestros días porque ha sido escogido como base para la especificación del lenguaje OWL DL.

Lenguajes de representación de conocimiento lógico

Web Ontology Language (OWL [BvHH+], en inglés *Web Ontology Language*) es un lenguaje de representación para la semántica declarativa propia de las ontologías y las bases de conocimiento de un sistema basado en conocimiento. El año 2002 este lenguaje se propuso como un estándar internacional para la Web Semántica por parte del *Consorcio de la red mundial* (W3C).

Existen tres dialectos distintos de OWL: OWL Lite, OWL DL y OWL Full. OWL Lite es el lenguaje más sencillo pero menos expresivo de los tres. OWL Full es el más expresivo pero menos asequible para el razonamiento automático, debido a su complejidad. OWL DL es el dialecto escogido para utilizarse en la Web Semántica y en multitud de otras aplicaciones que requieren razonamiento automático, debido a que ofrece un equilibrio adecuado entre expresividad y propiedades computacionales, resultando decidible y completo para la implementación de los llamados razonadores de DLs.

Este lenguaje añade algunas construcciones semánticas adicionales basadas en *DAML+OIL* [DARa] sobre RDF Schema (RDFS). *DAML* [DARb] y *OIL* [OTK] son lenguajes de representación de conocimiento en forma de marcos, siendo el primero para agentes software que operan en la Web Semántica y el segundo para modelos DL de la Web Semántica.

Resource Description Framework (RDF [W3Ca]) es un lenguaje para describir metadatos en Internet desarrollado por el Consorcio de la Red Mundial.

El modelo RDF se basa en que los distintos recursos de la web se describan mediante tripletas, expresiones con un sujeto, un predicado y un objeto. El sujeto hace siempre referencia al recurso que se va a describir, el predicado puede ser una propiedad una relación, y el objeto será, según el predicado anterior, o bien el valor de dicha propiedad que corresponde al recurso o bien la referencia a un segundo recurso que estaremos relacionando de esta forma con el primero.

SWRL [W3Cb] es un estándar del *Consorcio de la red mundial* para describir reglas de acuerdo al estándar *OWL*. Estas reglas, que actúan como implicaciones lógicas sobre el modelo formado por las ontologías y la base de conocimiento, aumentan la expresividad del lenguaje *OWL*. Este aumento de la expresividad tiene una consecuencia negativa, que consiste en que se pierde la decidibilidad de los razonadores convencionales de lógicas descriptivas. Es por eso que se ha identificado un subconjunto de SWRL de *reglas seguras para lógicas descriptivas* que permite aumentar también la expresividad de OWL pero sin llegar a perder características tan importantes para el razonamiento como la decidibilidad.

2.3.3. Creatividad y razonamiento computacional

La creatividad intelectual del ser humano es su capacidad para progresar individual y socialmente a través de la realización de nuevos descubrimientos que amplían su conocimiento personal o el de su cultura de forma significativa. Aunque la creatividad es un misterio que siempre ha intrigado al ser humano, que lleva siendo objeto de estudio formal desde los años 50 y que es un tema recurrente en proyectos de IA clásica, la comunidad científica dedicada a la Creatividad Computacional en realidad es de creación reciente, y los mecanismos que dan lugar a la creatividad son aún grandes desconocidos. Esta comunidad científica se ocupa de investigar los mecanismos de la creatividad, ya sea formulando teorías computacionales que tratan de explicarlos o desarrollando sistemas informáticos que tratan de simularlos.

Según Turner [Tur92], la creatividad es aquella habilidad que se pone de manifiesto cuando se resuelve un problema empleando los conocimientos que se poseen de una forma nueva. Esta investigación, por lo tanto, se justifica porque intenta incrementar el rendimiento de la creatividad humana en la resolución de problemas, o al menos porque intenta comprender mejor sus mecanismos; aunque sea la definición formal de sistema creativo objeto de mayor discusión aún que la de un sistema inteligente.

La creatividad computacional tiene cuatro componentes principales: el proceso, el producto, el autor y el entorno. El proceso es la forma en que se produce la creación. El producto es precisamente aquello que es creado. El autor, ya sea un sistema informático o un ser humano asistido por un sistema informático, hace referencia al creador y el historial de sus creaciones, incluyendo todos los procesos y los productos que en algún momento han estado vinculados con él. Finalmente el entorno es aquello que condiciona de alguna manera al proceso, pero sin llegar a determinarlo. Incluye también su propio historial de creaciones, al igual que el autor.

A la hora de estudiar el grado de creatividad de un sistema hay que tener en cuenta una peculiaridad de la evaluación en el campo de la Creatividad Computacional. La métrica para evaluar un sistema suele aplicarse sobre el producto en vez del proceso, porque este último suele ser mucho menos accesible.

Una de las referencias más importantes en el campo de la creatividad se encuentra en la Psicología, siendo muy difundida la obra de Margaret Boden *The Creative Mind* [Bod90]. Boden coincide con otros expertos en que la creatividad es uno de los aspectos de la inteligencia, y no una capacidad independiente a esta.

Según Boden, existen dos posibles perspectivas para estudiar la creatividad: la personal o psicológica, llamada *p-creatividad*, y la histórica, llamada

h-creatividad. Las medidas de la p-creatividad son relativas al autor, mientras que las medidas de la h-creatividad son relativas al entorno. Siguiendo estas ideas, Rafael Pérez y Pérez [PyPS04] concluye que la forma más razonable de medir la creatividad de un sistema informático pasa por evaluar su salida y compararla con la información presente en su base de conocimiento. Esta medida se denomina creatividad computerizada (*c-creatividad*.

Se tome una u otra perspectiva, la calidad del producto tiene dos componentes: su valor y su novedad. Normalmente cuando se experimenta en el laboratorio con sistemas creativos se miden el valor y la novedad del producto en términos p-creativos, es decir, en relación a otros productos que el sistema haya creado hasta el momento. Sin embargo para las aplicaciones finales se desea que demuestren un alto grado de valor y novedad en términos h-creativos, esto es, que resulten útiles y novedosos en el entorno real.

Con respecto al funcionamiento de la creatividad, Boden sugiere dos enfoques: la creatividad exploratoria, llamada *e-creatividad*, y la creatividad transformacional, llamada *t-creatividad*. Por un lado, el enfoque e-creativo contempla el proceso creativo como la combinación de fragmentos del producto en un determinado espacio de posibilidades, mientras que por otro, el enfoque t-creativo lo contempla como además con la posibilidad de aplicar transformaciones sobre ese espacio que faciliten encontrar combinaciones más adecuadas para formar el producto. La t-creatividad es, por tanto, una visión más amplia del proceso creativo, al igual forma que las obras más creativas de la Humanidad son aquellas que han supuesto el desplazamiento de un paradigma investigador hacia otro. Dichos descubrimientos de relevancia histórica son sólo transformaciones del espacio de búsqueda que permiten lanzar nuevos procesos e-creativos sobre un terreno virgen de posibilidades aún inexploradas.

Las técnicas de implementación de estos procesos son de lo más variadas, yendo desde los modelos matemáticos como los fractales a los sistemas de redes neuronales, pasando por el uso de reglas de producción, razonamiento basado en casos y algoritmos evolutivos. Estas técnicas admiten a su vez diferentes variantes, pudiendo generar productos a partir de un conjunto posible de elementos muy grande o a partir de un producto "semilla" que se establece como punto de partida, realizando evaluaciones del producto que está siendo generado al final del proceso o en iteraciones muy frecuentes durante el mismo.

Turner afirma en su tesis [Tur92] que la manera en que el ser humano resuelve problemas que requieren creatividad sugiere que existen distintos tipos de heurísticas para ello, algunas generales y otras específicas del dominio del problema concreto que se está resolviendo.

> This suggests that people may have a hierarchy of creativity heuristics, from general heuristics that apply weakly to a wide range of problems, down to specific heuristics that apply strongly to a single problem type. [Tur92] pag. 49

Es comprensible que siendo la narración un fenómeno importante de la experiencia intelectual humana, el estudio de la creatividad en ese dominio particular sea un tema recurrente de la investigación en IA.

A continuación se describen dos grandes paradigmas del razonamiento computacional, el razonamiento basado en reglas y el razonamiento basado en casos. Finalmente se añade otro apartado dedicado a los razonadores basados en DLs.

Razonamiento basado en reglas

El razonamiento basado en reglas es un modelo de proceso de resolución de problemas que trabaja aplicando una serie de transformaciones sobre el conocimiento inicial hasta conseguir dar con la solución. Se hicieron muy populares durante los años 70 y 80 en el desarrollo de los llamados sistemas expertos. Este modelo propone un proceso iterativo donde los hechos que figuran en la base de conocimiento, descriptores de un problema, se cargan en una memoria dinámica de trabajo donde sufren varias transformaciones de acuerdo a un conjunto de reglas hasta que llegar a un estado final donde, si todo ha ido bien, se encuentran los hechos que describen la solución deseada.

Las reglas suelen tener la forma de una implicación lógica, con un antecedente que debe cumplirse para que la implicación tenga sentido, y con un consecuente que deberá ser cierto en el caso de que el antecedente también lo sea. Esto en ocasiones puede entenderse en un sentido declarativo, considerando inconsistente la base de conocimiento que no cumple todas las implicaciones, aunque lo habitual es entenderlo en un sentido operacional, como ocurre en los sistemas de reglas de producción, donde las reglas se activan y provocan cambios en los hechos de la base de conocimiento.

Las reglas pueden actuar en dos sentidos, dando lugar a dos sistemas de razonamiento diferentes: *encadenamiento hacia adelante* y *encadenamiento hacia atrás*. En los sistemas de encadenamiento hacia adelante los hechos que describen el problema en su situación actual se comparan con los patrones que representan los antecedentes de todas las reglas, activando aquellas en las que los antecedentes se cumplen y ejecutándose a continuación sus consecuentes, que serán interpretados como modificaciones en los hechos de la memoria de trabajo. El orden en que dichas modificaciones se ejecutan depende de la estrategia de resolución de conflictos que se haya establecido previamente.

En los sistemas de encadenamiento hacia atrás los hechos iniciales también incluyen la solución deseada, posiblemente a falta de ligar algunas variables, y el problema consiste en averiguar si es posible demostrar la validez de la solución a partir de los hechos existentes en la base de conocimiento, si es necesario ligando las variables pendientes. Las reglas se activan en el orden inverso, es decir, si los consecuentes encajan con la solución hipotética que reside en la memoria de trabajo, se crean los correspondientes antecedentes imaginarios, confiando en que tras varias iteraciones se lleguen a obtener unos antecedentes imaginarios que coincidan, o puedan ser ligados, con los que forman parte de los hechos iniciales de la base de conocimiento.

Razonamiento basado en casos

El razonamiento basado en casos (CBR [RS89] del inglés *Case Based Reasoning*) consiste en un modelo de proceso general de resolución de problemas y aprendizaje inspirado en el modo en que los seres humanos resolvemos nuevos problemas reutilizando las soluciones satisfactorias que fueron aplicadas a problemas similares con anterioridad.

Las raíces del CBR están en el trabajo de Roger Schank y sus colegas de principios de los 80 [Sch82], en el área de la Psicología. El concepto fundamental de este tipo de razonamiento es el *caso* que se define, según Kolodner y Leake [KL96], como un fragmento de conocimiento que representa una experiencia que enseña una lección fundamental para conseguir los objetivos del razonador. Los casos normalmente representan un problema concreto y la solución que fue usada con éxito para resolverlo. La base de conocimiento de un sistema CBR suele llamarse simplemente base de casos.

Los sistemas CBR se presentan como una alternativa a los sistemas basados en reglas. Mientras que las reglas hacen referencia a normas generales bien definidas que se saben ciertas para el dominio de conocimiento sobre el que se trabaja, los casos capturan soluciones particulares para problemas concretos dentro de un dominio para el que no existe o no es fácil de conseguir una formalización completa. Es por ello que la técnica de CBR es adecuada cuando resulta más eficaz reutilizar soluciones previas que construirlas desde cero o averiguar un modo general de hallarlas, tratándose habitualmente de dominios de difícil análisis exhaustivo donde el aprendizaje tiene una función importante. Estos dominios de conocimiento deben cumplir un requisito importante para que la técnica de CBR funcione: si dos problemas son similares, sus soluciones, por norma general, también deben serlo.

Como requisito a la ejecución de un proceso CBR es necesario disponer de una base de casos inicial. La adquisición del conocimiento necesario para conseguir suficientes casos es el primer problema de la construcción de estos

sistemas, cuyo resultado afectará de manera importante al rendimiento del sistema. Por lo general es mejor disponer de una base de casos muy poblada que poco poblada, aunque los casos deben ser escogidos con cautela, ya que demasiados casos similares y mal organizados sólo servirán para hacer más costosa la búsqueda de los pocos que sean necesarios para el razonamiento.

Riesbeck y Schank [RS89] clasifican los casos en tres categorías. La primera categoría está formada por *casos proverbiales*, plantillas genéricas que resultan de utilidad para comenzar a resolver muchos problemas distintos, pero que carecen de información específica para precisar con exactitud la solución exacta a un problema particular. La segunda categoría son los *casos paradigmáticos*, muy específicos y con un gran nivel de detalle tanto en la descripción del problema concreto como en su solución, pero difíciles de reutilizar en problemas similares. Finalmente la tercera categoría está formada por los casos que estos autores llaman *historias*. Estos casos son de gran utilidad para un sistema CBR, al proporcionar suficiente detalle sobre el problema y su solución, como los casos paradigmáticos, pero también resumiendo información más general sobre la resolución de un tipo de problemas, como los casos proverbiales. Esto permite su reutilización en múltiples situaciones distintas.

El proceso CBR convencional se compone de cuatro etapas: recuperación, adaptación, evaluación y aprendizaje (a menudo expresadas en inglés como *retrieval*, *reuse*, *revision* and *retention*).

La etapa de recuperación consiste en, dado un nuevo problema a resolver, encontrar en la base de casos uno o más casos que resuelvan problemas similares. Para ello puede utilizarse un *índice de casos*, una *función de similitud* o una combinación de ambos. Los índices de casos sirven para organizar los casos según una estructura de datos que agrupe los casos con problemas similares entre sí, para evitar tener que realizar en esta etapa una búsqueda lineal exhaustiva sobre todos los casos de la base de casos. Las funciones de similitud sirven para comparar dos problemas y obtener un valor numérico que refleje lo similares que son, lo que permite escoger el caso o los casos más adecuados para resolver un problema dado. También es posible anidar varias etapas de recuperación, una tras otra, para refinar la búsqueda, usando índices de casos o funciones de similitud más precisas en cada iteración.

La etapa de adaptación consiste en, dado el problema a resolver y uno o más casos recuperados, generar una solución candidata para dicho problema. Para dar con la solución correcta existen muchas y muy diversas opciones de implementación. A menudo muchos sistemas realizan una *adaptación trivial*, que consiste en reutilizar directamente la solución de uno de los casos recuperados, dejando el éxito del proceso en función de los resultados de la etapa de recuperación. Otra opción es realizar una *adaptación manual*, de-

jando que sea el usuario el que genere la nueva solución usando para ello las soluciones recuperadas. Las alternativas automáticas pasan por realizar *transformaciones* sobre las soluciones anteriores hasta obtener una nueva. Estas transformaciones pueden ser sustituciones de unos fragmentos de la solución por otros, o pueden involucrar algoritmos mucho más complejos. Las transformaciones a realizar pueden clasificarse según se utilice o no la información del problema actual y de los problemas de los casos recuperados. Existe una alternativa algo menos convencional a la adaptación transformacional que se conoce como *adaptación constructiva*. Consiste en construir una solución al problema partiendo desde cero, usando cualquier otra técnica de resolución de problemas y complementando dicha técnica con las soluciones anteriores obtenidas en la etapa de recuperación. La forma de ir construyendo la solución puede ser iterativa, recurriendo cuantas veces sea necesario a repetir la etapa de recuperación para obtener distintas soluciones diferentes cuyos fragmentos sirvan para ir generando la solución candidata final.

La etapa de evaluación consiste en verificar si la solución candidata es una solución válida para el problema a resolver. Normalmente el mecanismo de evaluación debe ser ajeno al sistema CBR, pudiendo actuar como evaluador un ser humano o un sistema informático externo. Si la evaluación se realiza en el propio sistema CBR las métricas de evaluación en general serán heurísticas, o bien dependerán de algoritmos o simuladores que sólo pueden utilizarse para validar soluciones, no para generarlas; como por ejemplo sería la ejecución de otro ciclo de CBR distinto destinado a evaluar si la solución candidata es válida. Una evaluación negativa puede provocar una vuelta a la etapa anterior, en ocasiones con información que realimenta el sistema para mejorar la adaptación. La etapa de evaluación es opcional, siendo habitual que muchos sistemas la omitan, limitándose a dejar que sea el usuario el que la realice o aproveche la solución candidata a su manera.

Finalmente la etapa de aprendizaje consiste en decidir si el problema resuelto y su solución merecen figurar en la base de casos, añadiéndolos como un nuevo caso si así es. Por norma general aumentar automáticamente la base de casos mejorará el rendimiento del sistema, pero es importante asegurarse de que estos casos son correctos y suficientemente distintos a los ya existentes, teniendo en cuenta el volumen y la organización actual de la base de casos. Se pueden usar técnicas complementarias a este aprendizaje, como el "olvido" de casos que ya no tengan la relevancia que tenían en la base de casos inicial. De nuevo esta etapa puede dejarse en manos del usuario, si no es posible dar con heurísticas adecuadas que permitan enriquecer adecuadamente la base de casos. Aunque el aprendizaje automático es una de las características más significativas que el proceso CBR lleva integradas, muchos sistemas la omiten, al tratarse de una etapa compleja y dependiente del resultado obtenido de la

etapa anterior de evaluación.

El modelo convencional de un sistema CBR ha sido aplicado con éxito en varios dominios. Sin embargo, como expone Janet Kolodner [Kol93], este modelo está orientado a la reutilización de soluciones previas, evitando en la medida de lo posible la creación de soluciones novedosas. Esta característica es un impedimento muy serio a la hora de desarrollar aplicaciones que requieren soluciones que, además de válidas, sean diferentes a las soluciones propuestas anteriormente o que cumplan requisitos adicionales independientes de su validez. Como apunta Kolodner, el modelo CBR no es una técnica concreta de implementación, sino todo un paradigma de resolución de problemas inspirado en lo que conocemos sobre el funcionamiento de la mente humana; por lo tanto, también es posible alterar el modelo convencional para incorporar nuevas ideas sobre el modo en que la creatividad interviene en el proceso de resolución de problemas. Resulta incluso razonable pensar que los casos son una forma concreta de ver esas "semillas" iniciales de las que necesitan partir algunos sistemas creativos. Kolodner y su grupo desarrollaron sistemas de razonamiento creativo basado en casos como *Creative JULIA* [KP90], *CYRUS* [Kol83], *SWALE* [KLO86] o *CHEF* [Ham86]. En ellos a veces es necesario modificar el orden de las etapas del ciclo convencional incluyendo iteraciones o nuevas etapas dedicadas a la generación y evaluación de posibles soluciones creativas.

El razonamiento basado en casos intensivo en conocimiento (KI-CBR del inglés *Knowledge-Intensive Case Based Reasoning*) es un tipo particular de CBR donde se representa conocimiento de forma explícita en los casos; conocimiento que además se utiliza en las distintas etapas del ciclo, sobre todo en las de recuperación y adaptación.

Un sistema KI-CBR puede utilizar cualquier formalismo para representar el conocimiento. Una posibilidad que se popularizó pasada la década de los 90 es la de usar algún tipo de DL, dado que sus razonadores están especializados en clasificar de forma automática los conceptos y los individuos de la base de conocimiento, lo que resulta muy útil en algunas etapas del ciclo CBR. Poder reconocer el tipo de un individuo según sus roles y atributos para situarlo junto a otros que son instancias del mismo concepto facilita la recuperación de problemas similares a uno dado y también el aprendizaje de un nuevo caso, situándolo directamente en el lugar que le corresponda dentro de la base de conocimiento.

En el proyecto *COLIBRI* [DAGC03] (del inglés *Cases and Ontology Libraries Integration for Building Reasoning Infrastructures* se adopta un enfoque interesante que combina CBR, DLs y ontologías en un sólo sistema. Una ontología llamada *CBROnto* se utiliza para representar el metadominio de los casos de un sistema CBR, con sus problemas y soluciones. El ciclo de dicho

sistema CBR se modela utilizando las DLs para automatizar parte de las tareas de recuperación y adaptación [GAGCDAFC99] La ontología incluye un amplio conjunto de relaciones para los elementos de un caso, espaciales, temporales, o de otros muchos tipos, aunque la idea es que el sistema CBR sea independiente, debiendo ser su conocimiento extendido hacia los dominios de conocimiento concretos de la aplicación que va a ser implementada.

Razonadores basados en lógicas descriptivas

Pellet [Mina] es un razonador de lógicas descriptivas basado en *tableaux*. Se distingue por que tiene una comunidad muy activa de usuarios y desarrolladores, que incorporan resultados recientes de investigación en las características del razonador y ofrecen un API muy bien diseñado para integrarse fácilmente en cualquier aplicación. Se trata de un proyecto de código abierto, implementado íntegramente en *Java*. Alguna de las características particulares de *Pellet* está relacionada con la mejora en eficiencia de operaciones críticas como la validación automática de un modelo de DL. Esta operación, que a menudo suele resultar costosa, se puede realizar bajo ciertos supuestos de forma *incremental*, es decir, validando sólo aquella información que haya sido añadida al modelo desde la última vez que este fue validado.

RacerPro[RS], versión actual del antiguo *Racer*[HM03] es un razonador de lógicas descriptivas. Se distingue por su compacta y eficiente distribución. Se trata de un proyecto de código cerrado, cuyo ejecutable se distribuyó de forma gratuita hasta la versión 1.7.24 y hoy día es un producto comercial, aunque con licencias educativas gratuitas y renovables por tiempo indefinido.

2.3.4. Armazones orientados a objetos

Los armazones son una estructura de soporte para organizar y desarrollar aplicaciones informáticas en un cierto dominio, es decir, dedicadas a resolver una cierta familia de problemas similares. Básicamente el armazón surge al formalizar una metodología de desarrollo concreta y materializarla en forma de una arquitectura y unos componentes que modelan las entidades habituales del dominio objetivo, su estructura, sus relaciones y su comportamiento. Dicha arquitectura deberá ser instanciada en una aplicación concreta, a veces incluso es posible hacerlo de forma semiautomática, que pasará obligatoriamente a funcionar de acuerdo al armazón y a hacer uso de los componentes que este proporcione. Normalmente los armazones tiene arquitecturas altamente configurables, existiendo la posibilidad de prescindir de algunos componentes o incluso desarrollar algunos nuevos e incluirlos para luego utilizarlos desde la aplicación final.

El propósito de un armazón es facilitar el trabajo de los desarrolladores de aplicaciones, para que estos puedan dedicarse a los aspectos específicos de su aplicación en vez de concentrar el esfuerzo en resolver problemas más genéricos que ya han sido resueltos en aplicaciones similares a la suya.

Los armazones orientados a objetos usan el paradigma de la programación orientada a objetos para implementar el núcleo y los componentes de la arquitectura software que ofrecen. Habitualmente se proporcionan interfaces, clases abstractas y clases concretas pensadas para ser extendidas mediante herencia o composición por la aplicación final.

A continuación se describen los lenguajes de programación más habituales que se utilizan para implementar armazones orientados a objetos, añadiendo un apartado sobre el uso de armazones de tecnologías semánticas.

Lenguajes de programación orientada a objetos

Java [SM] es un lenguaje de programación de propósito general muy popular y muy consolidado tanto en el ámbito académico como en el empresarial. Este lenguaje, que se compila a un código intermedio que es ejecutado por una máquina virtual, resulta atractivo por su portabilidad entre distintas plataformas hardware, legibilidad y elegante aproximación a la orientación a objetos.

Como se pondrá de manifiesto en el siguiente apartado, *Java* es un lenguaje muy utilizado para desarrollar armazones, especialmente en el ámbito de las tecnologias de la Web Semántica, donde cuenta con una amplia comunidad de usuarios y un gran número de herramientas disponibles, la mayoría de código libre. Además *Java* cuenta con un entorno integrado de desarrollo de código libre muy popular y potente, llamado *Eclipse* [Ecl], y con otras herramientas de utilidad para la realización de pruebas automáticas (*JUnit* [Men]), el registro de errores (*Commons Logging* [Apaa] y *log4j* [Apab]) y la documentación (*Javadoc* [Sun]).

Una de las alternativas más potentes como lenguaje de programación orientado a objetos es *C++* [ISO]. Este lenguaje permite realizar desde programación genérica de alto nivel a programación de bastante bajo nivel, con lo que puede conseguirse un código compilado muy optimizado para la plataforma concreta donde se desee ejecutar.

Aunque también existe tecnología *C++* en el ámbito de la Web Semántica, cada vez es menos utilizado que *Java*.

Armazones de sistemas de razonamiento

Existen armazones prácticamente para cualquier tipo de aplicación informática. En este apartado se citan los más significativos que se han tenido presentes para la elaboración de este trabajo.

jCOLIBRI [GCDABT$^+$] es un armazón de código abierto para el desarrollo de sistemas CBR, evolución de las ideas surgidas en el proyecto *COLIBRI* mencionado en el apartado 4.1.3, que ahora se ven implementadas en *Java* en lugar de *Lisp*. Este armazón tiene dos versiones, una orientada al usuario no programador y otra más avanzada para usuarios programadores. El armazón además incorpora una interfaz de acceso a ontologías y bases de conocimiento llamado *OntoBridge* [RGDAGCS06], la cual a pesar de su simplicidad permite combinar un sistema CBR convencional con el uso de una representación intensiva en conocimiento.

Jena [HP], por otro lado, es uno de los armazones *Java* más populares para desarrollar aplicaciones basadas en tecnologías semánticas que utilicen estándares como RDF, RDFS y OWL. Ha sido desarrollado por los laboratorios de *Hewlett-Packard* y actualmente es un proyecto de código abierto.

Además de contar con APIs para RDF y OWL, con la posibilidad de leer y escribir ficheros en formato RDF/XML y en forma de tripletas, también cuenta con un sistema de almacenamiento persistente, motores para consultas, y un motor propio de inferencia basado en reglas. También puede comunicarse con un razonador de lógicas descriptivas externo e independiente gracias a la interfaz DIG, o acoplarse directamente con un razonador hecho en Java como *Pellet*.

DIG [BMC03] es una interfaz para la comunicación entre una herramienta cliente y un razonador de DLs. Esta interfaz ha sido estandarizada por el *DL Implementation Group*, que propone una implementación en formato XML. Su objetivo es desacoplar el contenido de los mensajes de las particularidades internas de cada herramienta o razonador de DL.

DIG 1.1 soporta dominios finitos como los números enteros y las cadenas alfanuméricas. Sin embargo tiene muchas limitaciones, por ejemplo al no ofrecer soporte para otros dominios finitos como el de los números con coma flotante, ignorando todas las restricciones que hagan referencia a esos dominios.

La alternativa en *C++* a herramientas como *Jena* es *Redland* [Bec], un conjunto de librerías que actúan como armazón para desarrollar aplicaciones que trabajen con el estándar *RDF*. Teniendo una comunidad de usuarios menor, la documentación y el número de ejemplos desarrollados no es comparable al de *Jena*.

Capítulo 3

Armazón para narración automática

Nunca sigas el camino recorrido,
porque sólo te conducirá a donde otros han estado.
Alexander Graham Bell

En este capítulo se presenta el grueso de la contribución de este trabajo de tesis, describiendo cual es la hipótesis de la que se parte, además del material y método de investigación utilizado durante el proceso, que parte de los objetivos propuestos en el apartado 1.4 y de todo lo estudiado en el capítulo 2.

La contribución principal de esta tesis es un armazón para el desarrollo de aplicaciones de narración automática, con un enfoque concreto sobre cómo debe almacenarse, organizarse y manipular el conocimiento necesario para generar automáticamente nuevas historias que satisfagan métricas preestablecidas de coherencia, valor y novedad. Para ello se construyen una serie de herramientas software, *DLModel* y *DLApplication*, que forman el núcleo del armazón, el cual se completa con un conjunto de componentes ontológicos básicos que es posible reutilizar durante el desarrollo de las aplicaciones, como se explicará al final del capítulo.

3.1. Hipótesis

Una vez revisado el estado de la cuestión a la luz de los objetivos de este trabajo, es posible plantear la hipótesis sobre la que parte esta investigación.

Antes de describir la hipótesis es necesario explicar cuales son las suposiciones sobre las que esta se apoya. En primer lugar, las aplicaciones de narración automática pueden considerarse válidas en términos empíricos, si

satisfacen tanto a sus usuarios como a sus desarrolladores. Por un lado se supone que los usuarios de estas aplicaciones están satisfechos con ellas si las aprueban de forma explícita tras haberlas usado, y por otro lado también se supone que los desarrolladores de estas aplicaciones están satisfechos con ellas si las aprueban de forma explícita tras haberlas desarrollado. En segundo lugar, las aplicaciones de narración automática también pueden considerarse válidas en términos formales, si satisfacen unos criterios de coherencia, valor y novedad preestablecidos.

La hipótesis de este trabajo de investigación es la siguiente: los objetivos propuestos en el apartado 1.4 pueden alcanzarse construyendo un armazón que permita desarrollar aplicaciones de narración automática para generar historias en un formato independiente a su presentación al público, teniendo que ser además fácilmente evaluables.

Este armazón está basado en componentes ontológicos específicos de dominio y algoritmos de generación basados en factores pseudoaleatorios, consistencia lógica, reutilización de conocimiento semántico y episódico. Como parte del armazón se proponen dos métodos de evaluación diferentes, uno empírico basado en cuestionarios que miden la satisfacción de los usuarios y los desarrolladores, y otro formal basado en inferencia y estadística básica, para medir la coherencia, el valor y la novedad de cada historia generada.

3.2. Material y método de investigación

El material de investigación consiste por una parte en documentos de la literatura científica disponible y por otro en herramientas teóricas y prácticas para el desarrollo de software.

El material disponible para la revisión bibliográfica ya fue descrito en el apartado 2.2.3 y las herramientas disponibles más interesantes fueron detalladas en el apartado 2.3.

Finalmente el material escogido para documentar esta investigación ha sido *LaTeX* [pro], *ArgoUML* [Tig] se ha utilizado para el diseño de diagramas, y también *JavaDoc* [Sun], *OWLDoc* [CO] y *Jambalaya* [CHI] se utilizan para documentar el código y las ontologías, estas últimas de forma textual y visual, respectivamente.

Para el desarrollo se ha utilizado *Java* [SM] como lenguaje de programación de todo el armazón, *Eclipse* [Ecl] como entorno integrado de desarrollo, *Jena* como implementación y conexión interna de las tecnologías semánticas, *Pellet* [Mina] como razonador y *Protégé-OWL* [GMF+02, CDF+] como editor de ontologías y bases de conocimiento, siendo OWL DL [BvHH+] y SWRL [W3Cb] los lenguajes escogidos para modelar el conocimiento declarativo de

los componentes del armazón.

El método de investigación que se sigue en esta investigación se compone de una serie de pasos: revisión, diseño, implementación, evaluación y documentación. Estos pasos se ejecutan en orden, aunque realizando varias iteraciones sobre la secuencia completa, de forma análoga a como funciona el proceso de desarrollo iterativo en Ingeniería del Software.

Para poder construir este armazón ha sido necesario analizar bien el área de la Narratología Computacional [GLRMP06], identificando y definiendo formalmente los problemas y encontrando teorías que planteen posibles soluciones para la representación y el procesamiento computacional de las historias. Se ha revisado de forma bastante exhaustiva el material bibliográfico para encontrar información relevante sobre la teoría de la narración automática. Como ya se ha visto en el capítulo 2.2, se han estudiado aquellos proyectos de investigación que plantean objetivos comparables con los de este trabajo.

El siguiente paso es el diseño de un modelo de narración automática que tenga en cuenta el tipo de autoría computacional, evaluación y validez que se han propuesto como objetivos del trabajo.En este paso se han utilizado las herramientas de documentación y diseño software mencionadas anteriormente, aunque para los primeros borradores se utilizaron notas manuscritas y ficheros de texto plano. Para satisfacer a los desarrolladores se decide hacer un uso intensivo de la programación orientada a objetos modular y la reutilización de componentes software, ambos principios básicos de la Ingeniería del Software, además de los principios de la Lógica y la Ingeniería Ontológica para la representación del conocimiento. Para satisfacer a los usuarios se contempla la posibilidad de que estos juzguen las historias generadas por la aplicación, además de que la propia aplicación posea alguna manera de filtrar las historias antes de presentarlas al público, eliminando aquellas que no resulten coherentes o suficientemente valiosas y novedosas.

El paso posterior al diseño es la implementación del armazón, de su núcleo software y de los componentes ontológicos básicos que lo completan, así como una aplicación de ejemplo que será de utilidad a continuación. Durante la implementación se han seguido los pasos habituales del ciclo de vida del software, con especial atención a la construcción e interconexión de las ontologías de los componentes.

El paso siguiente es evaluar el armazón, para lo que se utiliza la aplicación de ejemplo que lo instancia. Al evaluar la aplicación se puede discutir el funcionamiento del armazón, así como su validez y utilidad práctica, las ventajas y desventajas frente a otros sistemas revisados anteriormente, etc.

Finalmente el último paso es la documentación del trabajo usando el material de edición, diseño y desarrollo ya mencionado para preparar los ar-

tículos que se publicaron durante la investigación, para difundir los resultados entre los miembros de la comunidad científica internacional.

En los siguientes apartados se describe en profundidad el armazón propuesto. Primero se describe el núcleo del armazón y después el conjunto de componentes ontológicos de dominio que completan el armazón y que servirán para entender mejor el funcionamiento de la aplicación de ejemplo que se presentará en el capítulo 4.

3.3. Núcleo del armazón

La contribución principal de este trabajo de investigación es el armazón para el desarrollo de aplicaciones de narración automática. El *núcleo software* de este armazón se divide en dos grandes partes llamadas *DLModel* y *DLApplication*. La primera es una interfaz de acceso y manipulación interna de los modelos de lógicas descriptivas; de ahí las siglas DL del inglés *Description Logic*. La segunda es la parte general de la arquitectura del armazón que organiza todo lo necesario para que las aplicaciones desarrolladas utilicen la citada interfaz como modelo interno de conocimiento. Ambas partes del núcleo del armazón se describen con más detalle en los siguientes apartados.

3.3.1. *DLModel*

Para poder tratar de una manera uniforme y desacoplada las distintas tecnologías semánticas de acceso y manipulación de modelos de DL, se ha desarrollado una interfaz de programación de aplicaciones (API) especializada en esta tarea. Esta interfaz se conoce como *DLModel*[Peib].

El propósito de esta interfaz es facilitar el desarrollo de aplicaciones basadas en un modelo de conocimiento representado mediante DLs y en un modelo de razonamiento dotado de la capacidad de inferencia habitual que tienen los sistemas basados en DLs (validación, clasificación y reconocimiento de tipos).

Esta interfaz resulta sencilla de usar para el programador porque abstrae los detalles de bajo nivel de las distintas tecnologías semánticas que se utilicen internamente y permite usarlas como si todas ellas formaran un sistema de caja negra. *DLModel* abstrae únicamente los aspectos fundamentales del formalismo de las lógicas descriptivas, proporcionando al programador un acceso directo e intuitivo a las ontologías y bases de conocimiento implicadas, con los métodos imprescindibles para gestionar, consultar y razonar con los individuos de la base de conocimiento.

DLModel básicamente permite construir objetos que representan conceptos, roles e individuos. Estos objetos se componen siempre de un nombre local y otro objeto que indica el vocabulario de tipo TBox, RBox o ABox donde han sido declarados originalmente. De esta forma representan una referencia única dentro del modelo, el cual también es un objeto complejo de *DLModel* y posee todos los métodos de acceso y modificación disponibles para el programador.

A través de esos métodos que tiene el modelo el programador puede realizar consultas sobre los conceptos, roles e individuos que ha sido previamente cargados en él, así como sobre muchas de sus relaciones lógicas; se puede comprobar su existencia en el modelo, recorrer las relaciones familiares en la polijerarquía de conceptos o roles, averiguar los tipos de un individuo, obtener todos los individuos relacionados con otro individuo, etc. El programador también puede crear nuevos individuos cuyas referencias ha construido previamente, con la seguridad de que a medida que utiliza los métodos de consulta y modificación de la base de conocimiento, serán lanzados automáticamente los procesos de razonamiento considerados "estándar" en sistemas basados en DLs, como son la validación completa del modelo, la clasificación de todos los conceptos o el reconocimiento de todos los tipos a los que corresponde cada individuo.

La libre modificación de un modelo basado en DLs, concretamente el añadir nuevos individuos a la base de conocimiento o el relacionarlos con otros individuos, puede dar lugar a inconsistencias lógicas en el mismo. Estas inconsistencias, salvo casos excepcionales, no permiten razonar con el modelo y lo dejan completamente inservible desde un punto de vista práctico. Es por eso que en *DLModel* todos los métodos que pueden causar dicha inconsistencia devuelven un valor booleano, que será cierto si la modificación ha podido realizarse con éxito y falso si el modelo se ha dejado intacto porque la modificación habría causado inconsistencias. Aunque algo costosa, esta estrategia preventiva es la que permite a las aplicaciones usar modelos de DL con total seguridad de que no van a crearse inconsistencias.

DLModel tiene una implementación en *Java* que recibe el nombre de *jDLModel*. Actualmente funciona a modo de envoltura del armazón de tecnologías semánticas *Jena* y el razonador *Pellet*, con quien Jena puede comunicarse directamente o de manera remota a través de *DIG*.

Entre otras utilidades, *jDLModel* incluye un modelo de ejemplo sobre el que se pueden realizar varias decenas de pruebas automáticas *JUnit* para validar el funcionamiento de todos los métodos de la API. Además existe un registro de errores *log4j* de *Commons Logging* para dejar rastro en fichero de todas las acciones que realiza el programador al usar la API.

3.3.2. *DLApplication*

Para desarrollar aplicaciones basadas en uno o más modelos de DL se utiliza un núcleo software con las operaciones mínimas imprescindibles para usar la tecnología del apartado anterior de una forma homogénea y ordenada. Este núcleo, la parte más genérica del armazón propuesto en este trabajo, se conoce como *DLApplication*[Peia].

DLApplication da forma a una arquitectura, asociada a las ideas generales de esta tesis doctoral, útil para el desarrollo de aplicaciones con componentes ontológicos que hacen uso de *DLModel* para resolver problemas complejos que involucran razonamiento sobre individuos de varios dominios de conocimiento distintos.

Dicha arquitectura es modular, está basada en componentes fácilmente mantenibles y reutilizables, y resulta lo suficientemente genérica como para servir de base para muchas y muy diversas aplicaciones. Como se ha mencionado anteriormente el armazón propuesto en este trabajo se compone de dos partes, el núcleo y los componentes. Estos componentes, que se detallarán más adelante, encapsulan la semántica declarativa y operacional relativa a un dominio de conocimiento, formalizando la semántica declarativa de acuerdo a una ontología y la semántica operacional de acuerdo a un paquete orientado a objetos con métodos de acceso y manipulación para cualquier base de conocimiento que se estructure según dicha ontología.

En la Figura 3.1 se muestra el diagrama de despliegue de una aplicación *DLApplication*. Los elementos de color blanco representan el núcleo software, que incluye el modelo de DL de la aplicación, y los componentes de dominio, con sus ontologías y ficheros de configuración asociados. Los elementos de color gris representan todas esas otras partes de la aplicación independientes del armazón propuesto, como la interfaz gráfica de usuario, el razonador de DL que puede estar ubicado en una máquina distinta de la que ejecuta la aplicación, el resto de la web, etc. Estos elementos de color gris en el diagrama sirven para recordar que una aplicación desarrollada con *DLApplication* puede completar su funcionalidad y la de la API de *DLModel* con otros métodos y otras herramientas diferentes, no necesariamente relacionadas con DLs.

La estructura de la semántica operacional de los componentes se establece aquí, en *DLApplication*. Esta consiste en varios interfaces, clases abstractas y clases concretas destinadas a facilitan la implementación del código correspondiente a cada componentes. Básicamente un componente se divide en cuatro paquetes importantes orientados a objetos: model, modelHandler, exception y test.

El paquete model representa el modelo de lógicas descriptivas subyacen-

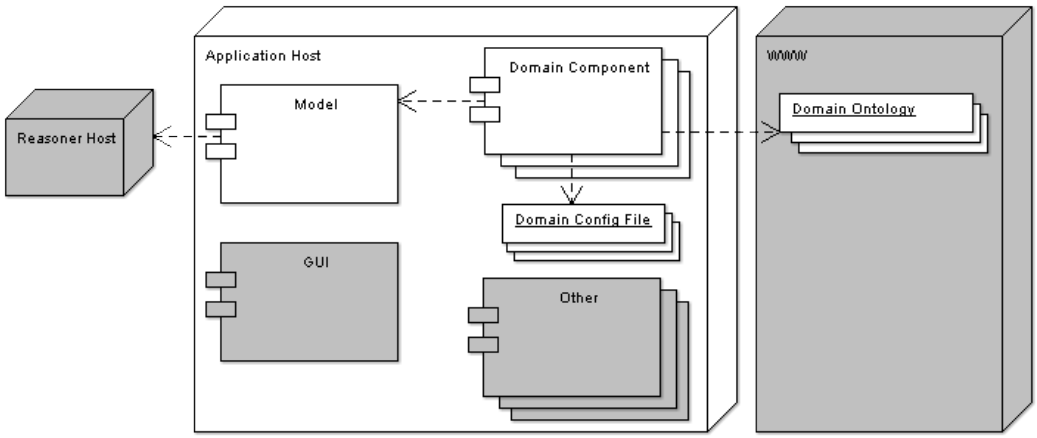

Figura 3.1: Diagrama de despliegue de una aplicación *DLApplication*

te, visto según un dominio de conocimiento concreto, Contiene métodos para obtener referencias a los conceptos, roles y atributos de la ontología de dominio asociada al componente, así como para crear nuevos individuos de tipos normalizados en ese dominio. Estos conceptos, roles y atributos, al igual que los métodos de acceso tienen nombres únicos con respecto al espacio de nombres del componente, de modo que no hay inconveniente en que se utilicen los mismos nombres locales que aparecen en otros componentes.

El paquete modelHandler representa el manejador que otros componentes y también la propia aplicación tendrán a su disposición para manipular el modelo a través de la vista específica de dominio de este componente. Contiene métodos de *acceso*, *modificación*, *creación* y *destrucción* específicos para los individuos del dominio que modela el componente y todas sus relaciones.

El paquete exception y el paquete test son respectivamente las excepciones que lanza el componente y las baterías de pruebas que pueden ejecutarse para probar su funcionalidad.

Una utilidad interesante de *jDLApplication* es *DLApplication Wizard*, una pequeña aplicación que, dado un modelo de DL, genera automáticamente todas las interfaces y todas las clases del paquete model, exception e incluso parte de modelHandler, para todos los dominios que aparezcan en el modelo. Esto es una ayuda muy valiosa para el desarrollador, con la que evita tener que escribir manualmente código trivial de acceso a los conceptos, roles e individuos de las ontologías.

En la Figura 3.2 se muestra el diagrama de clases de un componente según la arquitectura de *DLApplication*. Los elementos de color blanco son total o parcialmente generados automáticamente por DLApplication Wizard, mientras que los elementos de color gris representan los interfaces y clases

donde el desarrollador del componente tendrá que programar la semántica operacional específica del mismo.

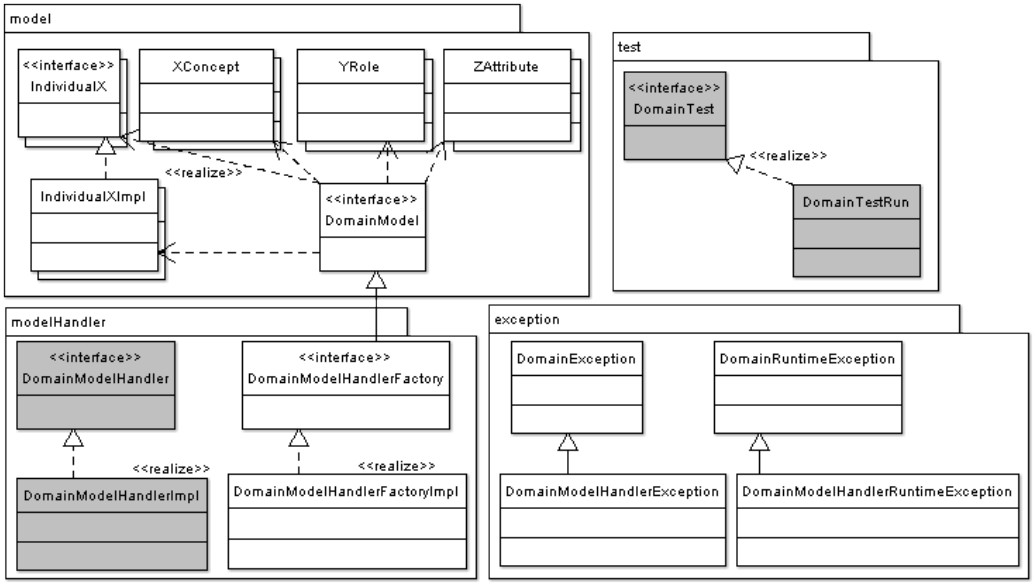

Figura 3.2: Diagrama de clases de un componente según *DLApplication*

DLApplication tiene una implementación en *Java* que recibe el nombre de *jDLApplication* y que actualmente es una extensión de *jDLModel*, incluyendo también soporte para que se realicen las pruebas automáticas *JUnit* y registro de errores *log4j* de *Commons Logging* en cada componente.

3.4. Componentes básicos del armazón

Los componentes ontológicos de dominio son la otra mitad del armazón propuesto, que conecta con el núcleo software del mismo, lo que quiere decir que siguen la arquitectura de *DLApplication* y por lo tanto utilizan la interfaz de *DLModel* para acceder a sus modelos internos de DL.

Estos componentes concentran toda la semántica tanto declarativa como operacional que el armazón ofrece como punto de partida para sus aplicaciones. No pretenden ser soluciones universalistas, sino que están orientados a resolver los problemas del campo de la narración automática, siempre buscando un compromiso razonable entre lo genérico y lo específico, para que resulten útiles por sí mismos, pero a la vez reutilizables en más de una aplicación diferente.

Para diseñar las ontologías que modelan el conocimiento de cada componente se utilizan unos principios metodológicos extraídos del proyecto *GOP*

descrito en el apartado 2.3.1; concretamente los principios de economía cognitiva, razonamiento metonímico y categorización en niveles básicos. El razonamiento metonímico en realidad está implícito en el razonamiento con DLs y la categorización en niveles básicos puede verse perfectamente como una referencia a la jerarquía conceptual normalizada que se exige para cada componente.

Los conceptos, en el contexto de este armazón, deben pertenecer exclusivamente a uno de estos tres tipos: *normales*, *no normales*, y *definidos*.

Los conceptos normales son conceptos primitivos según lo estudiado en el apartado 2.3.2, es decir, no poseen definiciones de condiciones necesarias y suficientes, aunque sí pueden poseer declaraciones de condiciones necesarias. Todos los conceptos normales además forman parte de una única monojerarquía principal que estructura todo el conocimiento; es decir, los conceptos normales se distinguen porque sólo tienen un padre.

Los conceptos no normales también son conceptos primitivos, pero además de tener un padre normal pueden tener otro padres, creando así una polijerarquía. Para distinguirlos de los conceptos normales en las figuras donde aparezcan jerarquías conceptuales, se les llama conceptos "tipados", asignándoles en muchos casos el nombre de su padre normal con el prefijo Typed-. Los conceptos no normales no pueden tener descendientes que sean conceptos normales.

Por último, los conceptos definidos en *DLApplication* se corresponden con los conceptos definidos mencionados en el apartado 2.3.2, por lo que poseen tanto declaraciones de de condiciones necesarias como definiciones de condiciones necesarias y suficientes. Los conceptos definidos no pueden tener descendientes normales ni no normales.

Los tipos de los individuos reciben un nombre, según sea el concepto del que son instancia. De esa forma los conceptos normales actúan como tipos emphnormalizados, los conceptos no normales como tipos *no normalizados* y los conceptos definidos como tipos *inferidos*.

Todos los individuos deben tener exactamente un concepto normal como tipo directo, esto es, un tipo normalizado. Al acto de asertar el hecho de que un individuo sea instancia de un tipo normalizado se le llama "normalizar" un individuo, una idea que surge evidentemente de la normalización de ontologías mencionada anteriormente. Todos los individuos de la base de conocimiento deben estar normalizados, aunque además de su correspondiente tipo normalizado pueden tener más tipos no normalizados o inferidos.

Algunos conceptos normales, habitualmente los que son muy generales, tienen la restricción de no deber ser asertados como tipos directos de un individuo, declarando que son la unión entre todos sus conceptos hijos y que por lo tanto debe asumirse que en realidad el tipo directo normalizado de sus

individuos siempre será el de alguno de sus hijos; a estos conceptos normales se les denomina conceptos *abstractos*. Los conceptos no normales sólo pueden asertarse como tipos directos de individuos una vez están normalizados. Los conceptos definidos directamente no pueden ser asertados como tipos directos de ningún individuo, por lo que solo el sistema puede asignar, de acuerdo al razonamiento automático de los sistemas basados en DLs, dichos tipos inferidos.

Una vez vistas las consideraciones para implementar la semántica declarativa de los componentes, pueden hacerse algunos comentarios sobre cómo debe implementarse la semántica operacional. Una aclaración importante es que los individuos que se crean a través del paquete model son siempre individuos normalizados. De hecho a la hora de añadir métodos al paquete modelHandler existe la restricción de operar solamente con individuos normalizados, accediendo siempre a los individuos en función de su tipo normalizado, también para crearlos o destruirlos, y evitando la modificación de dicho tipo salvo en el caso de que se sustituya por otro más específico o porque se añadan tipos no normalizados a un individuo ya normalizado.

En general, al relacionar individuos entre sí, se deben evitar formar ciclos entre individuos normalizados, ya que son esos individuos los que más se recorren habitualmente, y por tanto la existencia de ciclos es peligrosa, ya que podría haber algoritmos en el código de un componente que no sepan detectarlos y que por tanto caigan en un bucle infinito.

Para poder desarrollar aplicaciones de narración automática es necesario suministrar, además de una implementación del núcleo software del armazón, un conjunto de componentes ontológicos básicos ya implementados que modelen el conocimiento y el comportamiento de los dominios más usados en este tipo de aplicaciones.

Las aplicaciones que instancian *DLApplication* se comportan como sistemas basados en conocimiento, siendo dicho conocimiento formalizado principalmente en las ontologías y bases de conocimiento de sus componentes. El criterio fundamental con el que se construyen estos componentes es precisamente relativo al conocimiento que modelan: no sólo la semántica declarativa es dependiente de cada dominio de conocimiento que se pretenda modelar, sino también la propia semántica operacional. Por eso, como punto de partida para instanciar una aplicación de narración automática, se facilita un armazón ya interconectado por relaciones de *especialización* y *combinación* entre los conceptos y roles de sus correspondientes ontologías y relaciones de herencia entre las clases de sus correspondientes paquetes software. Las relaciones de especialización entre componentes hacen básicamente que conceptos de componentes inferiores refinen las declaraciones de conceptos de componentes superiores, heredando sus restricciones, mientras que las rela-

ciones de combinación entre componentes crear nuevos conceptos que de alguna manera conectan en su definición conceptos de componentes superiores. La dotación básica de componentes del armazón puede ser reutilizada total o parcialmente, e incluso extendida con otros componentes específicos de la aplicación, para crear el conjunto definitivo que se utilizará en el producto final.

En la Figura 3.3 se muestra la jerarquía de componentes básicos del armazón propuesto. Como puede verse en el diagrama se trata de una polijerarquía que admite herencia múltiple, que se implementa gracias a la flexibilidad de los mecanismos de herencia de las ontologías OWL DL y los interfaces *Java*. Para todos los componentes de color blanco se ofrece una implementación básica como punto de partida para las aplicaciones a desarrollar. Estos componentes representan la narración y la simulación que tiene lugar en una historia "abstracta", aún no presentada ante el público, e incluyen dominios más generales como la fábula, el discurso, el espacio o el tiempo. Para los componentes de color gris y la base de conocimiento no se ofrecen implementaciones, ya que su desarrollo, además de costoso, está estrechamente ligado a cada aplicación específica que se desarrolle. Estos componentes representan el propósito de la historia, ya sea un dominio pedagógico, ideológico o de otro tipo; y también el conocimiento de los *metadominios* que puedan hacer falta para hacer funcionar la aplicación, estructuras de datos como índices, punteros, etc. que no representan individuos concretos de ningún dominio pero son útiles para procesar el conocimiento.

La división del armazón en varios componentes diferentes favorece el diseño en dos aspectos fundamentales. Por un lado el armazón se vuelve más mantenible al poderse realizar pruebas y busqueda de errores de forma más acotada. Por otro lado es posible desarrollar bases de conocimiento que contengan conocimiento incompleto, como por ejemplo estructuras narrativas sin contenido de simulación o historias sin propósitos concretos asociados, información que resulta plenamente reusable y compatible con la de otras bases de conocimiento, gracias a la forma en que está diseñado el armazón. Todo esto es posible gracias a una organización en capas de componentes relacionados por una relación de especialización o combinación, según lo explicado anteriormente, de manera que los componentes situados más abajo en la jerarquía o bien refinan los conceptos de los superiores, heredando sus restricciones, o bien incorporan nuevos conceptos que conectan conceptos de componentes superiores de alguna manera.

Dentro de los componentes básicos no se incluye ninguno dedicado a la presentación de la historia. Aunque esta es una cuestión importante de cara al producto final, en este trabajo la presentación es una cuestión accesoria e independiente del núcleo de una aplicación de narración automática.

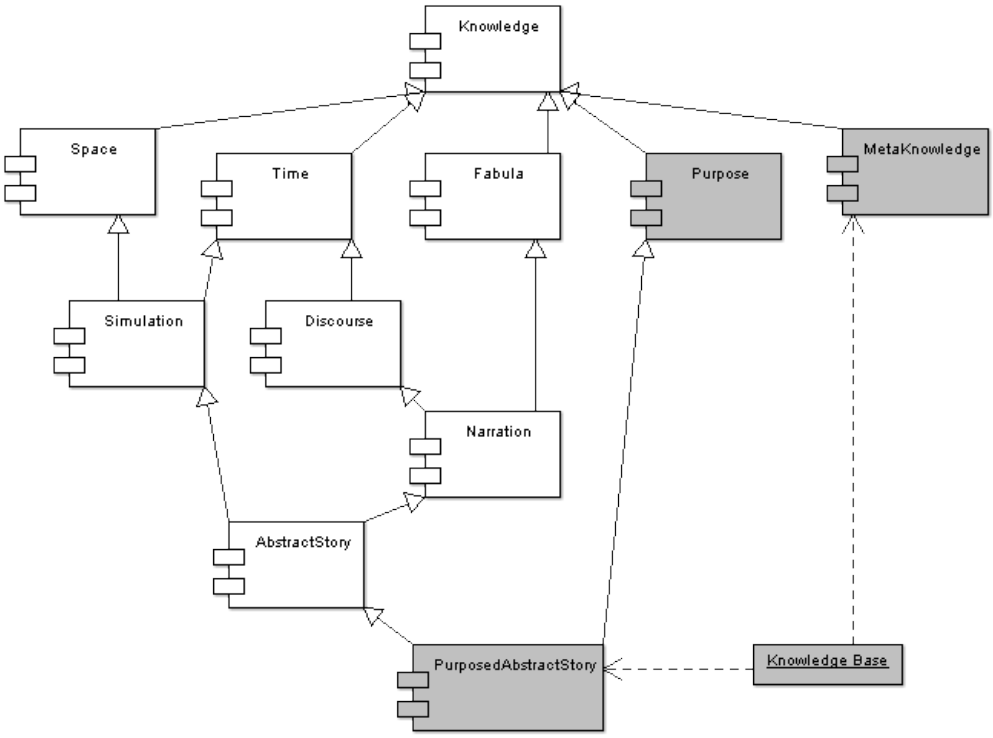

Figura 3.3: Jerarquía de componentes básicos del armazón

A continuación se describen estos componentes básicos, aunque debido a lo extenso de su contenido, la documentación exhaustiva y su implementación pueden encontrarse a través del apéndice A.

3.4.1. *Knowledge*

El componente *Knowledge* es un componente ontológico básico para el desarrollo de aplicaciones de narración automática utilizando *DLApplication*.

Este componente contiene la semántica declarativa y operacional del modelo general de conocimiento propuesto como raíz de todos los demás componentes que formarán parte de las aplicaciones finales. Los conceptos de este componente son *abstractos*, lo que quiere decir que no se contempla que pueda haber individuos que los instancien como tipos directos. Se considera que estos conceptos serán los ancestros de *nivel superior* de los todos los tipos de individuos que poblarán las bases de conocimiento de las aplicaciones finales desarrolladas mediante el armazón propuesto. Los conceptos de este dominio tienen una jerarquía *poco fragmentada*, lo que significa que los componentes que lo especialicen no lo harán de forma parcial sino especializando todos o casi todos sus conceptos. Este componente no tiene dependencias con ningún otro componente.

Una vista general de la jerarquía conceptual de este componente se muestra en la Figura 3.4.

Los únicos conceptos normales de este componente son Thing, Context y Element. Thing es la raíz de todos los conceptos normales que figuran en la jerarquía conceptual principal que recorre todos los componentes del modelo. En el concepto Thing se declara que todos sus individuos deben ser de tipo Context o de tipo Element. El primero, Context, representa a aquellos individuos que actúan como individuo principal o contexto que *cosifican* o *reifican*, es decir, representan en sí mismos toda una realidad perteneciente a un dominio de conocimiento concreto. El segundo, Element, representa a todos aquellos individuos que forman parte de un contexto y se relacionan entre sí para representar su estructura interna.

Los conceptos no normales son TypedThing, TypedContext y TypedElement. TypedThing es la raíz de todos los conceptos no normales que figuran en las distintas jerarquías conceptuales de los demás componentes del modelo, y como excepción a los demás conceptos no normales, sólo tiene un padre: Thing. Los otros dos, sin embargo, tienen un padre normal, Context en el caso de TypedContext y Element en el caso de TypedElement, más otro padre que es además antecesor común de todos los conceptos no normales: TypedThing. Estos conceptos no normales permanecen "vacíos", es decir, no tienen hijos en este dominio porque todos los conceptos normales son abstractos. Sin em-

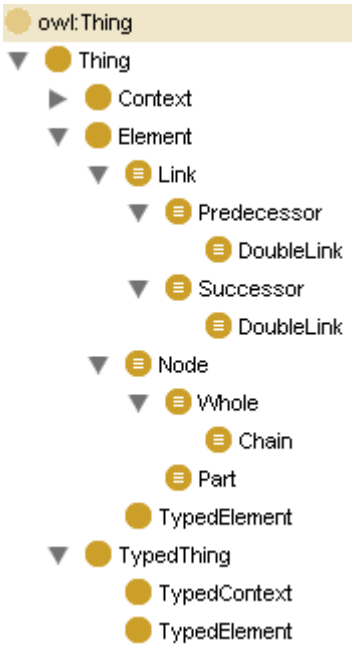

Figura 3.4: Jerarquía conceptual del componente *Knowledge*

bargo en los componentes que especializan este, aparecen conceptos normales instanciables y también conceptos no normales concretos que actúan como tipos complementarios para los individuos de cualquier base de conocimiento.

La jerarquía de roles de este componente se muestra en la Figura 3.5.

Los roles de este componente son el vocabulario esencial para la semántica declarativa del resto de componentes. Los roles más generales son refersTo, isElementIn, isContextIn y sus respectivos roles inversos, que habitualmente no describimos para no hacer redundante la explicación de los roles de cada componente. El rol refersTo es transitivo y conecta un individuo con todos a los que se puede llegar mediante relaciones de referencia. Este rol tiene roles hijos más específicos, como directlyRefersTo, precedes, hasPart y dependsOn, estando sus inversas ubicadas bajo isReferredBy. El rol directlyRefersTo representa una referencia directa de un individuo a otro. El rol precedes y su inversa también son transitivos y sirve para conectar individuos en secuencia. Tiene dos versiones más específicas, directlyPrecedes e isThePreviousTo, la primera para conectar directamente un individuo como predecesor de otros, y la segunda para hacer lo mismo pero con restricciones funcionales e inversamente funcionales que obligan a que sólo pueda existir un único predecesor y un único sucesor para el individuo que usa este rol. El rol hasPart tiene también una versión directa para conectar dos individuos sin transitividad,

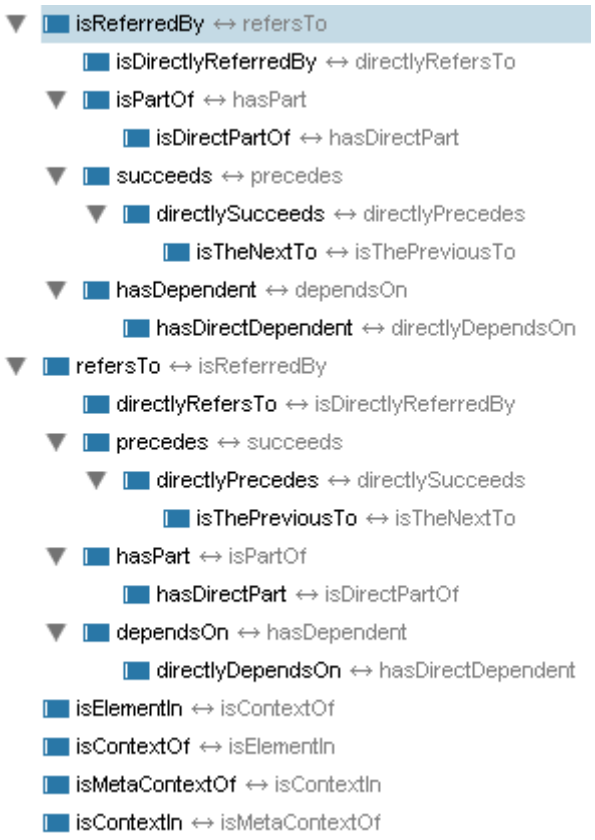

Figura 3.5: Jerarquía de roles del componente *Knowledge*

llamada hasDirectPart, ocurriendo lo mismo con el rol dependsOn que representa las dependencias lógicas de un individuo con respecto a otros. El rol isElementIn se utiliza para conectar un elemento con uno o más contextos a los que pertenece, y el rol isContextIn se utiliza para conectar un contexto con uno o más metacontextos (contextos que pueden alojar otros contextos en su interior) donde se encuentra ubicado.

Volviendo a los conceptos, los restantes son definidos y se utilizan para reconocer automáticamente aquellos individuos que cosifican eslabones de una lista o nodos de un árbol. Los primeros, de tipo Link, se definen como aquellos individuos que son o bien de tipo Predecessor o de tipo Successor, esto es, que tienen al menos una relación mediante el rol precedes o el rol succeeds, respectivamente, con otro individuo. Los segundos, de tipo Node, se definen como aquellos individuos que son o bien de tipo Whole o de tipo Part, esto es, que tienen al menos una relación mediante el rol hasPart o el rol isPartOf, respectivamente, con otro individuo. Estas estructuras, en forma de lista o árbol doblemente enlazado para facilitar su recorrido, son las estructuras más primitivas de conocimiento que se contemplan para los componentes del armazón. Tienen además algunos casos especiales, concretamente DoubleLink, aquel eslabón que tiene tanto sucesores como predecesores directos, y Chain aquel nodo cuyas partes son todas eslabones de una lista.

Con respecto a las operaciones sobre individuos normalizados, puede verse un resumen de la API de este componente en el Cuadro 3.1. El acceso, la creación y la destrucción se realizan exclusivamente sobre los individuos de tipo normalizado Knowledge, aunque la modificación permite añadir cualidades a un individuo, y en general los métodos tienen en cuenta los tipos inferidos del individuo afectado para realizar cambios en la base de conocimiento en consecuencia.

Acceso: A individuos Context y todos sus individuos Element conectados
Modificación: De individuos Context y sobretodo de individuos Element, básicamente añadiendo tipos no normalizados o estableciendo relaciones entre elementos
Creación: Tanto de individuos Context como Element, aunque hay métodos que permiten crear directamente árboles o listas, con parámetros para añadir los elementos que los forman
Destrucción: Tanto de individuos Context como Element, aunque si resultan ser individuos de tipo Chain se puede destruir la cadena completa en cascada

Cuadro 3.1: Resumen de la API del componente *Knowledge*

3.4.2. *Space*

El componente *Space* es un componente ontológico básico para el desarrollo de aplicaciones de narración automática utilizando *DLApplication*.

Este componente contiene la semántica declarativa y operacional del modelo de conocimiento propuesto para representar el espacio físico, sus distintos lugares y las entidades que los ocupan. Los conceptos de este componente son *concretos*, pueden ser instanciados como tipos directos, aunque estan diseñados para actuar como ancestros de *nivel superior* de los lugares y entidades de componentes más específicos. Dichos conceptos tienen una jerarquía *poco fragmentada*. Este componente especializa al componente *Knowledge*.

Una vista general de la jerarquía conceptual de este componente se muestra en la Figura 3.6.

Los conceptos normales de este componente son World, Location y Entity. World especializa el concepto Context del componente *Knowledge*, representando por tanto un contexto espacial, o mundo físico estático que sirve como referencia común para los elementos que existan en él. Location especializa el concepto Element de *Knowledge*, representando concretamente una localización física, una parcela de espacio delimitada y concreta. Entity también especializa el concepto Element de *Knowledge* y representa una entidad física del mundo.

Los conceptos no normales que sirven para enriquecer a los individuos de este dominio son los descendientes de TypedWorld, TypedLocation y TypedEntity, quienes a su vez especializan a los conceptos normales World, Location y Entity. Los conceptos no normales también especializan de un modo un otro a la raíz de los conceptos no normales: TypedThing de *Knowledge*. Esta herencia múltiple se produce en todos los conceptos no normales de todos los componentes, aunque en las figuras dicha polijerarquía tan recurrente no se muestra desplegada para ahorrar espacio en el documento y ganar en claridad. Los descendientes de TypedWorld, son Real y Virtual, tipos disjuntos que pueden asignarse a un mundo. Los descendientes de TypedLocation son Artificial y Natural, tipos disjunto de una localización. El hijo de TypedEntity es el concepto LifeBeing, que a su vez es especializado por SexualBeing, concepto que es declarado como la unión de sus dos hijos disjuntos Male y Female, indicando que todo ser vivo y sexuado tiene que ser obligatoriamente macho o hembra. También se proporciona un tercer hijo de SexualBeing llamado HumanBeing que obviamente no es disjunto de sus hermanos, y de hecho hereda la declaración de su padre, lo que le obliga también a repartir todos sus individuos divididos entre dos sexos.

La jerarquía de roles de este componente se muestra en la Figura 3.7.

Los roles de este componente son isWorldOfLocation, isWorldOfEntity, has-

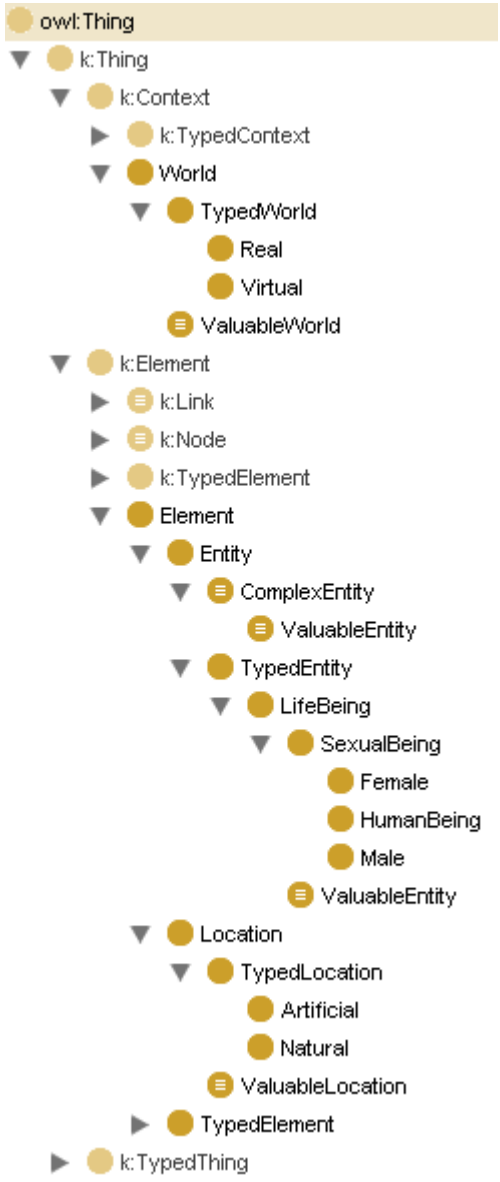

Figura 3.6: Jerarquía conceptual del componente *Space*

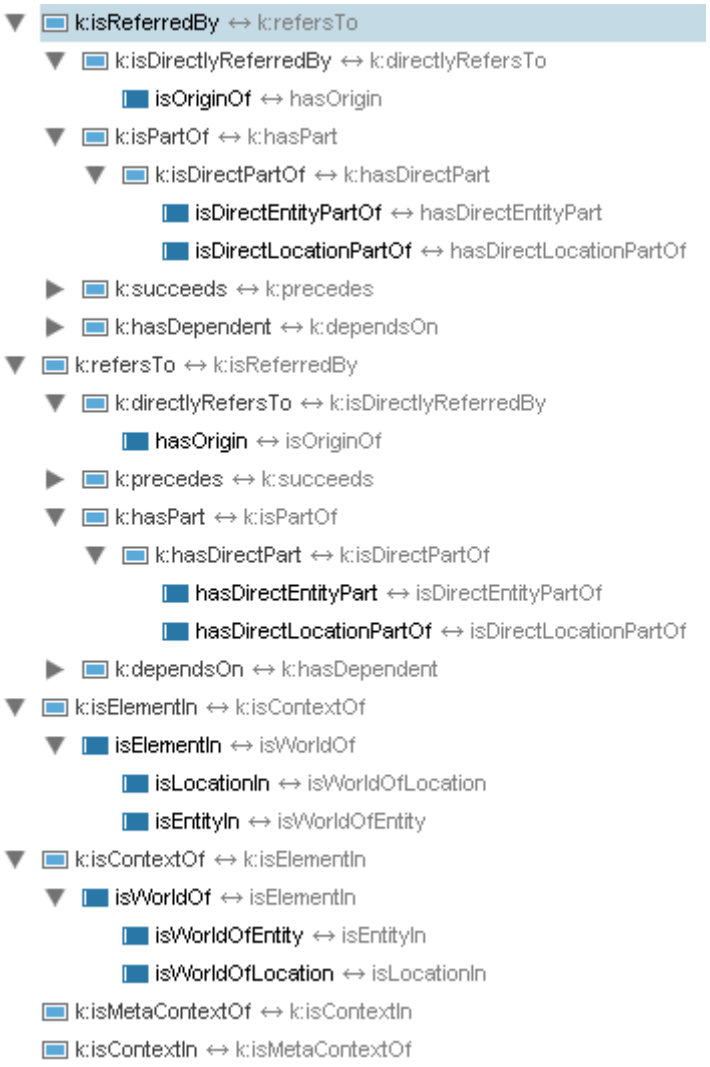

Figura 3.7: Jerarquía de roles del componente *Space*

DirectLocationPartOf, hasDirectEntityPartOf, hasOrigin y sus correspondientes inversas. Los roles isWorldOfLocation e isWorldOfEntity relacionan al mundo con las entidades y localizaciones que se encuentran en él. Los roles hasDirectLocationPartOf y hasDirectEntityPartOf permiten representar entidades complejas, formadas por subentidades y localizaciones complejas, formadas a su vez por sublocalizaciones. Finalmente el rol hasOrigin sirve para relacionar una entidad con su lugar de origen, no necesariamente la localización donde se encuentra, ya que la ubicación física de una entidad es en realidad una cuestión temporal que se trata en un componente más específico.

Volviendo a los conceptos, los restantes son definidos y son ComplexEntity, ValuableWorld, ValuableLocation, y ComplexEntity. El concepto ComplexEntity realiza una función auxiliar, definiéndose como el conjunto de aquellas entidades que tienen subentidades como partes. Los conceptos ValuableWorld, ValuableLocation y ComplexEntity sirven para identificar mundos, localizaciones y entidades valiosas, siempre según el criterio formal adoptado por el desarrollador de este componente, y de cara a ser útil en la evaluación de las historias generadas por las aplicaciones finales tal y como se explica en el apartado 3.6.2. El concepto ValuableWorld en este componente se define, usando restricciones de cardinalidad, como aquel mundo que tiene al menos dos localizaciones y tres entidades con origen en dichas localizaciones, además de contar tanto con localizaciones como con entidades de tipos inferidos "valiosos". La definiciones propuestas en los componentes básicos son siempre sencillas y tratan de trasmitir la idea de que los contextos serán valiosos cuando, entre otras razones, cuenten con algunos elementos valiosos en su interior. El concepto ValuableLocation se define como aquella localización que es origen de al menos dos entidades, existiendo alguna entidad valiosa entre ellas, algún ser vivo de sexo masculino y algún ser vivo de sexo femenino. El concepto ValuableEntity se define como aquella entidad de tipo no normalizado LifeBeing y tipo inferido ComplexEntity.

Con respecto a las operaciones sobre individuos normalizados, puede verse un resumen de la API de este componente en el Cuadro 3.2. El acceso, la creación y la destrucción se realizan exclusivamente sobre individuos normalizados, aunque la modificación permite asignar tipos no normalizados a dichos individuos.

3.4.3. *Time*

El componente *Time* es un componente ontológico básico para el desarrollo de aplicaciones de narración automática utilizando *DLApplication*.

Este componente contiene la semántica declarativa y operacional del modelo de conocimiento propuesto para representar el tiempo, en forma de una

Acceso: A individuos World, Location, Entity y todos sus individuos conectados **Modificación:** Principalmente de individuos Location y Entity, añadiendo tipos no normalizados o haciendo más específico su tipo normalizado **Creación:** De individuos World, Location y Entity **Destrucción:** De individuos World, Location y Entity, ocurriendo una destrucción en cascada de todos los elementos asociados a cada mundo que se destruye

Cuadro 3.2: Resumen de la API del componente *Space*

secuencia de instantes e intervalos. Los conceptos de este componente son *concretos*, pueden ser instanciados como tipos directos, actuando como referencias temporales a las que estarán conectadas otros individuos para reflejar que sufren cambios a lo largo del tiempo. En este sentido el componente es considerado de *nivel inferior*, ya que lo habitual es que no sea especializado por ningún otro componente. Además sus conceptos tienen una jerarquía *poco fragmentada*. Este componente especializa al componente *Knowledge*.

Una vista general de la jerarquía conceptual de este componente se muestra en la Figura 3.8.

Los conceptos normales de este componente son Timeline, Duration, y Event. Timeline representa la referencia de todos los elementos que pertenecen a una misma línea temporal. Duration representa la duración temporal de un evento, siendo el único hijo disponible en este componente básico la duración más sencilla posible: el instante de tiempo o individuo de tipo Instant. Event representa el evento temporal, el elemento clave de este componente, que irá relacionado con una línea temporal a través de su duración.

Los conceptos no normales de este componente son descendientes de TypedTimeline y TypedEvent. Los descendientes de TypedTimeline son Fast y Slow, tipos disjuntos que sirven para distinguir entre líneas temporales según dos escalas distintas. Los descendientes de TypedEvent son Complex y Simple, tipos disjunto de un evento.

La jerarquía de roles de este componente se muestra en la Figura 3.9.

Los roles de este componente son isTimelineOfElement, hasDuration, textsfdirectlyPrecedes y sus respectivas inversas. El rol isTimelineOfElement relaciona a la línea temporal con los eventos y las duraciones que se sitúan en ella. El rol hasDuration relaciona cualquier individuo de tipo Event con la duración que le corresponde. El rol directlyPrecedes conecta una duración tras otra, creando la secuencia temporal que servirá después para ubicar que evento ocurre antes o después que otro, etc.

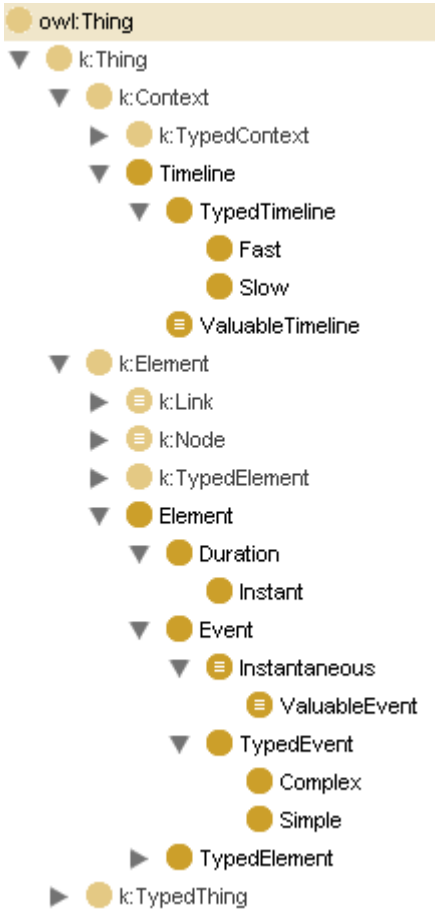

Figura 3.8: Jerarquía conceptual del componente *Time*

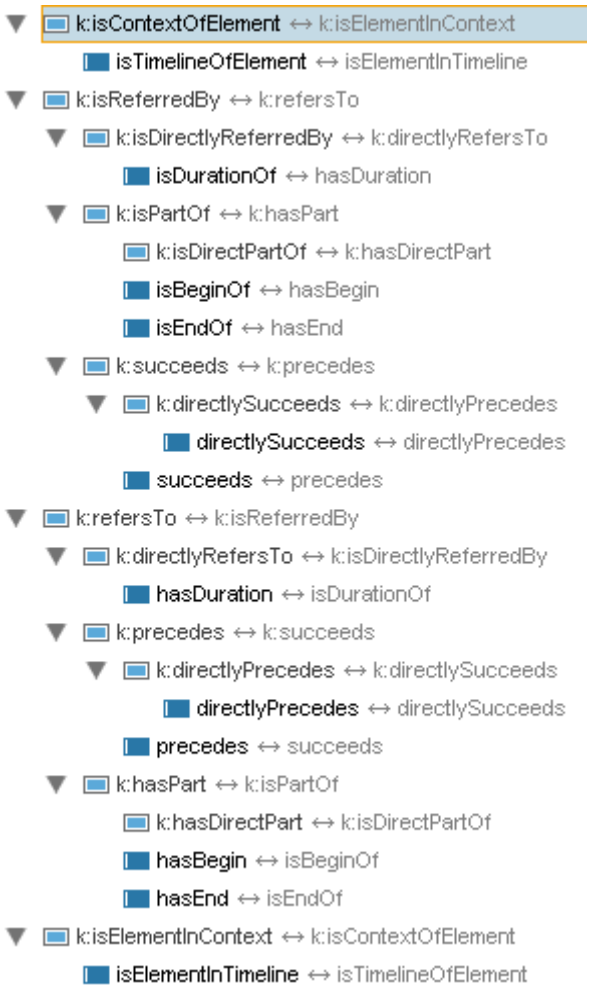

Figura 3.9: Jerarquía de roles del componente *Time*

Volviendo a los conceptos, los restantes son definidos y son Instantaneous, ValuableTimeline y ValuableEvent. El concepto Instantaneous es auxiliar, y se define un evento instantáneo, aquel cuya duración es únicamente un instante; el caso más habitual para los eventos que proponen para modelar en este componente básico. Los conceptos ValuableTimeline y ValuableEvent representan líneas temporales y eventos valiosos según el criterio propuesto en este componente. El concepto ValuableTimeline en este componente se define como aquella línea temporal que contiene al menos tres eventos temporales. El concepto ValuableEvent se define como aquel evento de tipo inferido Instantaneous que tiene una duración que es predecesora o sucesora con la duración de otro proceso, es decir, formando parte al menos de una cadena temporal de dos eventos.

Con respecto a las operaciones sobre individuos normalizados, puede verse un resumen de la API de este componente en el Cuadro 3.3. El acceso, la modificación, la creación y la destrucción se realizan exclusivamente sobre individuos normalizados.

Acceso: A individuos Timeline, Duration, Event y todos sus individuos conectados
Modificación: Principalmente de individuos Duration y Event
Creación: De individuos Timeline, Duration y Event, pudiendo crear cadenas enteras de estos dos últimos
Destrucción: De individuos Timeline, Duration que no tengan ningún evento asociado (ocurriendo un reorganización de las cadenas temporales que se vean interrumpidas) y Event

Cuadro 3.3: Resumen de la API del componente *Time*

3.4.4. *Simulation*

El componente *Simulation* es un componente ontológico básico para el desarrollo de aplicaciones de narración automática utilizando *DLApplication*.

Este componente contiene la semántica declarativa y operacional del modelo de conocimiento propuesto para representar la simulación del mundo ficticio de una historia, con los procesos físicos que se producen en él. Los conceptos de este componente son *concretos* y facilitan un modelo lógico de la realidad suficientemente general y sencillo. Lo más habitual es que cada aplicación especialice este componente con el tipo de agentes, objetos y lugares que van a aparecer en sus historias. Este componente es considerado de *nivel medio*, ya que puede o no ser especializado por otro componente.

Además sus conceptos tienen una jerarquía de *fragmentación media*, lo que quiere decir que este componente modela muchas facetas de la realidad distintas y por tanto lo esperado es encontrar componentes que sólo especialicen algunas de esas facetas. Este componente especializa al componente *Space* y al componente *Time*, al tiempo que los combina.

Una vista general de la jerarquía conceptual de este componente se muestra en las Figuras 3.10 y 3.11.

Los conceptos normales de este componente son Simulation, Agent, Object, State y Process, aunque son también directamente utilizados algunos conceptos heredados, especialmente del componente *Space*. Simulation es la referencia a la simulación de los eventos que ocurren en el mundo ficticio de una historia. Agent representa un agente de la simulación, esto es, una entidad capaz de provocar procesos. Existe un concepto que especializa Agent llamado IntelligentAgent que representa a aquel tipo de agente capaz de provocar procesos mentales que solo se dan en seres dotados de inteligencia. Object representa una entidad material, un objeto de la simulación, siendo Item una especialización para aquellos objetos de comportamiento sencillo y reactivo, generalmente manipulables y portables por un agente inteligente. State representa el estado de un agente, objeto o localización en un instante concreto de la simulación. Tener individuos distintos para un elemento y para todos los estados por los que dicho elemento va pasando a lo largo de la simulación permite poder asignar tipos no normalizados de dos forma distintas. La primera forma es *permanente*, asignando el tipo al elemento propiamente dicho para representar una propiedad intrínseca del mismo. La segunda forma es *temporal*, asignando el tipo a un estado del elemento, representando una propiedad transitoria del individuo que sólo es cierta en ese estado, el cual tendrá una duración concreta en la línea temporal de la simulación. Hay diferentes descendientes de State, según el elemento de la simulación al que se aplican, existiendo LocationState y EntityState, que a su vez distingue entre AgentState y ObjectState. Process representa un proceso de simulación física ocurriendo en un momento y un lugar determinado del mundo ficticio e involucrando a una serie de estados de agentes, objetos y localizaciones.

Los conceptos no normales de este componente son descendientes de TypedSimulation, TypedAgent, TypedObject, TypedState y TypedProcess. Los descendientes de TypedSimulation son Long y Short, según el alcance de los procesos que ocurren en la simulación. Los descendientes de TypedAgent son Collective y Standalone, según el agente sea claramente individual o emerja de un conjunto similar a como actúa un enjambre o una manada. Los descendientes de TypedObject son Dynamic y Static según se trate de un objeto dinámico como una llama o estático como una roca. Los descendientes de TypedState son TypedEntityState y TypedLocationState. Bajo TypedEntityState

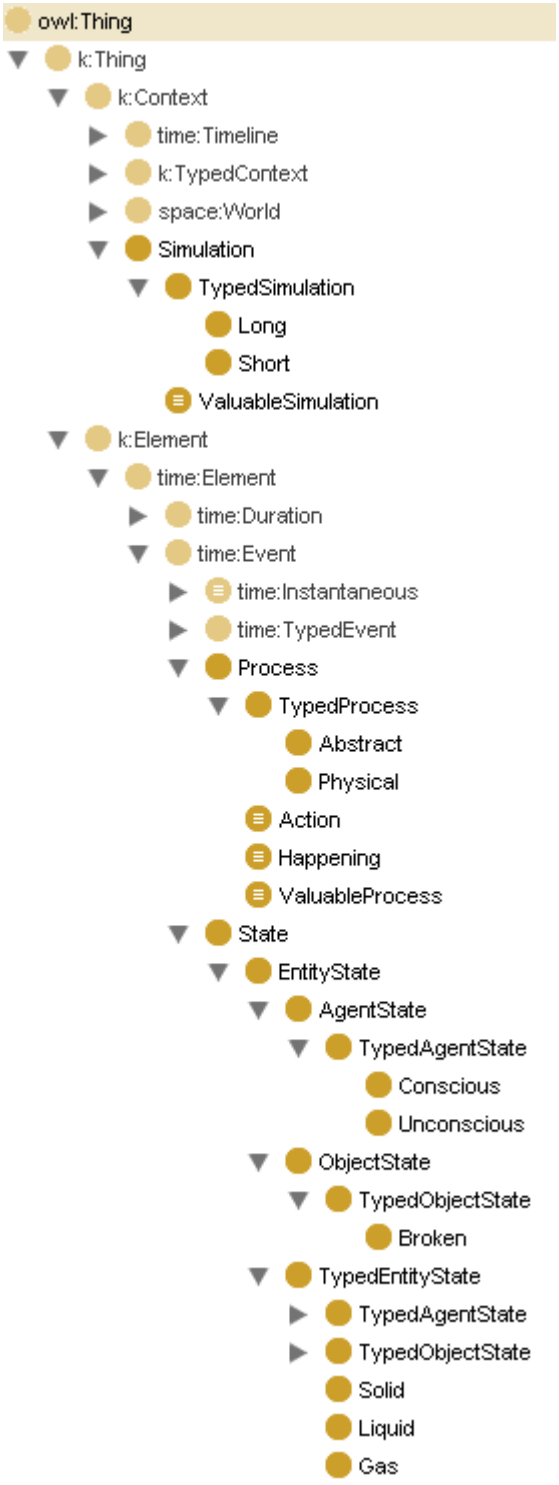

Figura 3.10: Jerarquía conceptual del componente *Simulation* (1/2)

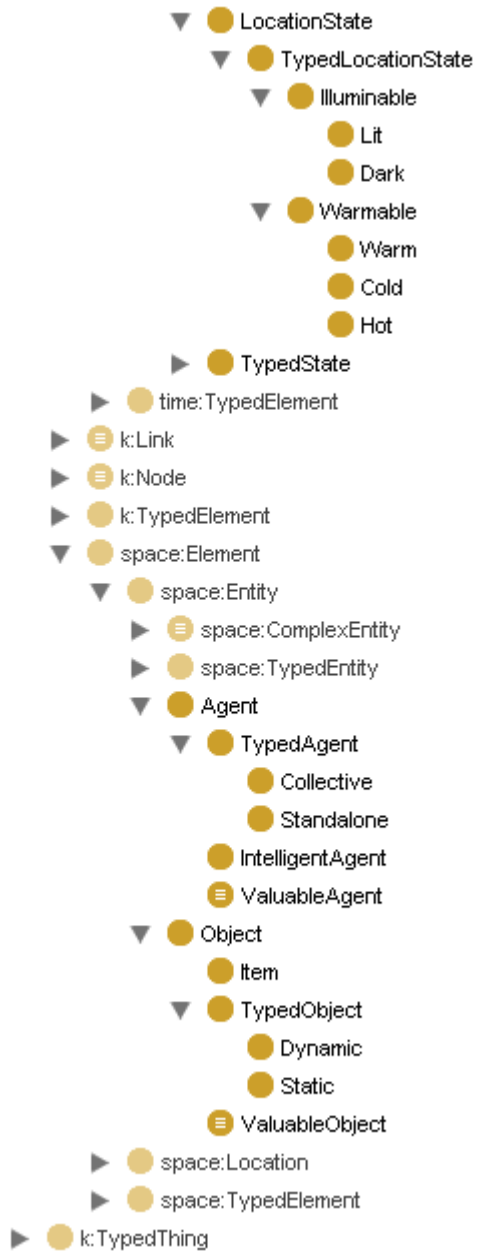

Figura 3.11: Jerarquía conceptual del componente *Simulation* (2/2)

hay tres conceptos disjuntos: Solid, Liquid y Gas, además de los dos hijos TypedAgentState y TypedObjectState. Bajo TypedAgentState hay dos conceptos disjuntos: Conscious y Unconscious, y bajo TypedObjectState hay sólo un concepto llamado Broken que representa el estado de los objetos que están rotos. Bajo TypedLocationState hay conceptos para representar la temperatura de una localización y si esta se encuentra iluminada o no, propiedades que claramente se asocian con el estado de una localización y no con la localización en sí, porque son mutables. Finalmente los descendientes de TypedProcess son Physical y Abstract, y se utilizan para distinguir los procesos físicos de otros más abstractos.

La jerarquía de roles de este componente se muestra en las Figuras 3.12 y 3.13.

Los roles de este componente son isSimulationOfWorld, isSimulationOfTimeline, isWorldOfAgent, isWorldOfObject, isTimelineOfProcess, isTimelineOfState, hasState, hasAgentState, hasDirectEntityState, hasIndirectEntityState, hasLocationState, directlyPrecedes y sus respectivas inversas. Los roles isSimulationOfWorld e isSimulationOfTimeline conectan a la simulación con el mundo y la línea temporal que tiene asociados. Los roles isWorldOfAgent e isWorldOfObject especializan el rol isWorldOfEntity del componente *Space* para los agentes y los objetos de la simulación, mientras que los roles isTimelineOfProcess, isTimelineOfState hacen lo mismo con el rol isTimelineOfEvent del componente *Time* para los procesos y los estados. El rol hasState relaciona una entidad o una localización con sus estados temporales. El rol hasAgentState relaciona proceso con el estado de un agente que lo ha ejecutando en ese momento de la simulación. El rol hasDirectEntityState relaciona un proceso con el estado de una entidad, agente u objeto, que participa en él de manera directa o activa. El rol hasIndirectEntityState relaciona un proceso con el estado de una entidad, agente u objeto, que participa en él de manera indirecta o pasiva. El rol hasLocationState relaciona un proceso con el estado de una localización donde este se está produciendo en ese momento de la simulación. Finalmente el rol directlyPrecedes y su inversa sirven para ordenar cronológicamente los procesos.

Volviendo a los conceptos, los restantes son definidos y son Action, Happening, ValuableSimulation, ValuableProcess, ValuableAgent y ValuableObject. Los conceptos Action y Happening son auxiliares, y definen dos tipos disjuntos de procesos: las acciones, en las que un agente participa como sujeto, y los sucesos, donde no hay agentes involucrados. El concepto ValuableSimulation representa una simulación valiosa según la siguiente definición: simulaciones cuyo mundo es valioso y al menos tiene dos agentes, cuya línea temporal es valiosa y tiene al menos tres procesos, o simulaciones donde ambas condiciones se cumplen. El concepto ValuableProcess se define como un proceso de

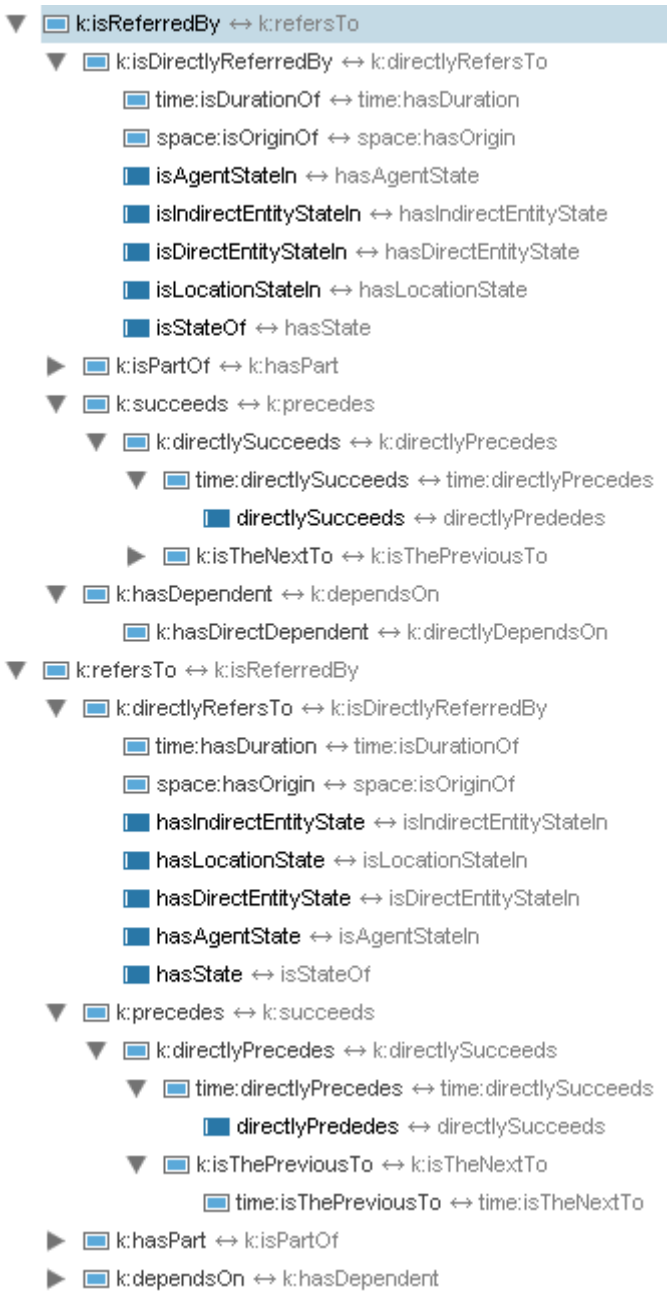

Figura 3.12: Jerarquía de roles del componente *Simulation* (1/2)

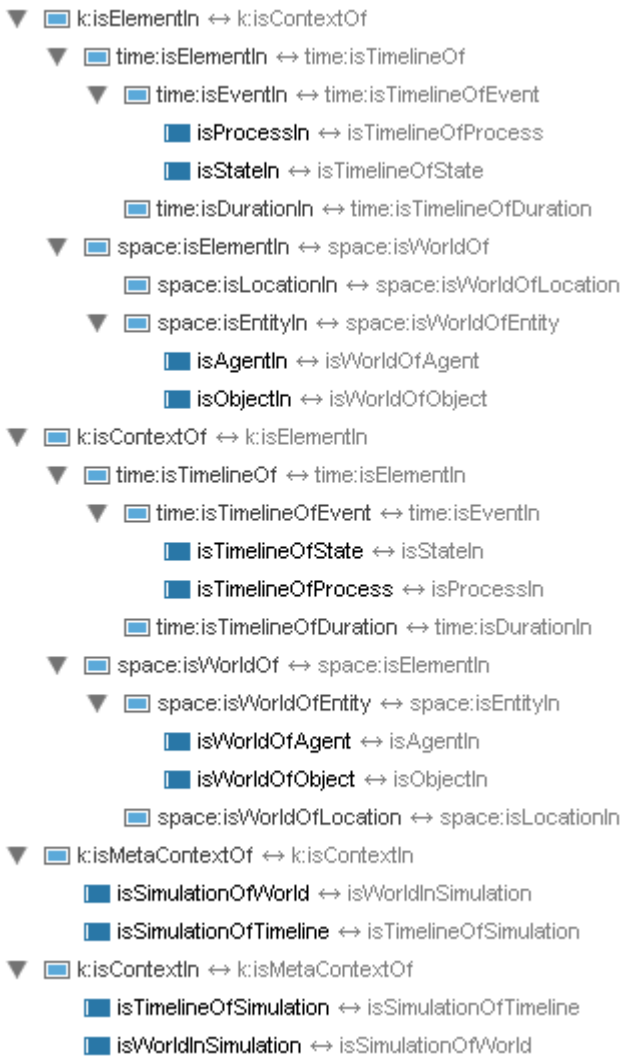

Figura 3.13: Jerarquía de roles del componente *Simulation* (2/2)

tipo Action producidos sólo por estados de agentes valiosos, o un proceso de tipo Happening donde sólo intervienen de forma directa estados de agentes u objetos valiosos. El concepto ValuableAgent se define como un agente con al menos tres estados que produce al menos un proceso y también participa como entidad directa en uno de ellos. El concepto ValuableObject se define como un objeto con al menos dos estados y que participa como entidad directa en algún proceso.

Con respecto a las operaciones sobre individuos normalizados, puede verse un resumen de la API de este componente en el Cuadro 3.4. El acceso, la creación y la destrucción se realizan exclusivamente sobre individuos normalizados, pudiendo modificarse individuos en el sentido de añadirles tipos no normalizados de un componente más específico.

Acceso: A individuos World, Agent, Object, State y Process y todos sus individuos conectados, que serán devueltos según el tipo que les corresponda
Modificación: De individuos Agent, Object y especialmente State y Process, siendo reordenados los procesos respetando el orden temporal y teniendo en cuenta la entidad que hay detrás de cada estado que interviene en un proceso
Creación: De individuos Agent, Object, State y Process, pudiendo crear cadenas temporales de estos últimos, y habiendo métodos para crear e incluir automáticamente estados para los agentes y objetos que participan en la creación de un proceso
Destrucción: De individuos Agent, Object, State y Process, ocurriendo un reorganización de las cadenas de procesos que se vean interrumpidas

Cuadro 3.4: Resumen de la API del componente *Simulation*

3.4.5. *Fabula*

El componente *Fabula* es un componente ontológico básico para el desarrollo de aplicaciones de narración automática utilizando *DLApplication*.

Este componente contiene la semántica declarativa y operacional del modelo de conocimiento propuesto para representar las fábulas de las historias, sus personajes y las funciones que estos desempeñan. Los conceptos de este componente son *concretos* y modelan la narratología básica expuesta en el apartado 3.4.6, implementando las ideas de Chatman y Crawford según las cuales no existe ni espacio ni tiempo físico en este nivel, sino meras estructuras causales. De todas formas las aplicaciones pueden especializar este

componente con un modelo de fábula más específico que les permita crear individuos más adecuados para el tipo de aplicación concreto de que se trate. Este componente es considerado de *nivel medio*, ya que puede o no ser especializado por otro componente narrativo. Además sus conceptos tienen una jerarquía *poco fragmentada*. Este componente especializa al componente *Knowledge*.

Una vista general de la jerarquía conceptual de este componente se muestra en la Figura 3.14.

Los conceptos normales de este componente son Fabula y Element, siendo Setting, Existent y CharacterFunction los conceptos hijos de este último, todos ellos elementos no físicos que sólo existen en la fábula, según el modelo narrativo de Chatman revisado en el apartado 3.4.6. Fabula es la referencia a la fábula de una historia concreta. Setting representa un escenario de dicha fábula. Existent representa un existente de la fábula, que puede ser de dos tipos: Character o Prop, según se trate de un personaje de la fábula o un mero elemento accesorio. CharacterFunction representa una función de personaje, cada uno de los eslabones de la cadena causal que constituyen la parte principal de la fábula.

Los conceptos no normales de este componente son descendientes de TypedFabula, TypedSetting, TypedCharacter, TypedProp y TypedCharacterFunction. Los descendientes de TypedFabula están basados en los tres tipos de conflictos que pueden dar lugar a una historia, según explicaba McKee en el apartado : Cosmic, Social y Personal. Los descendientes de TypedSetting son Calm y Dangerous según la sensación de peligro que el escenario provoque en los personajes. Los descendientes de TypedCharacter son Primary, Secondary, Tertiary, según el personaje sea primario, secundario, o terciario, existiendo un hijo particular del concepto Primary que representa al protagonista mediante el tipo Protagonist. Los descendientes de TypedProp son Bonus o Malus, según el significado que tenga el accesorio para el protagonista. Los descendientes de TypedCharacterFunction son Main o Satellite, según la función de personaje sea principal o una función satélite que sirve para rellenar la fábula.

La jerarquía de roles de este componente se muestra en las Figuras 3.15 y 3.16.

Los roles de este componente son isFabulaOfCharacterFunction, isFabulaOfCharacter, isFabulaOfProp, isFabulaOfSetting, isDirectCauseOf, hasExistent, hasSetting, hasCharacter y sus respectivas inversas. Los roles isFabulaOfCharacterFunction, isFabulaOfCharacter, isFabulaOfProp, isFabulaOfSetting relacionan una fábula con sus funciones de personaje, personajes, accesorios y escenarios, respectivamente. El rol isDirectCauseOf y su inversa relacionan las funciones de personaje entre sí para formar la cadena causal de una historia. Los roles

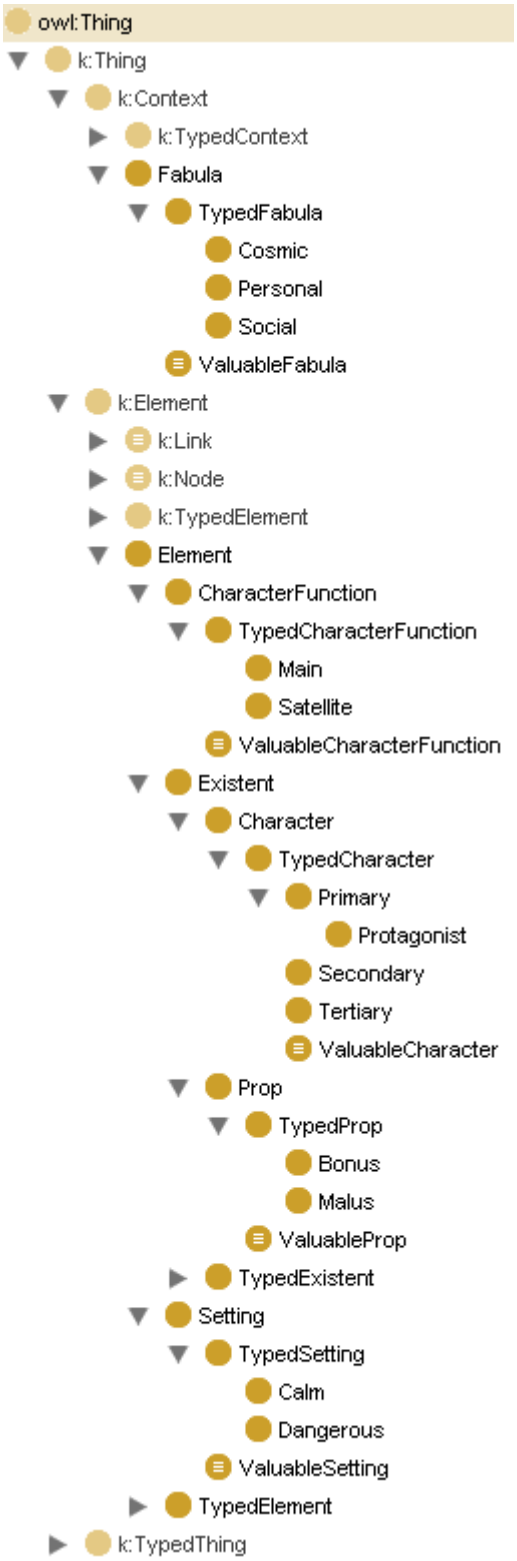

Figura 3.14: Jerarquía conceptual del componente *Fabula*

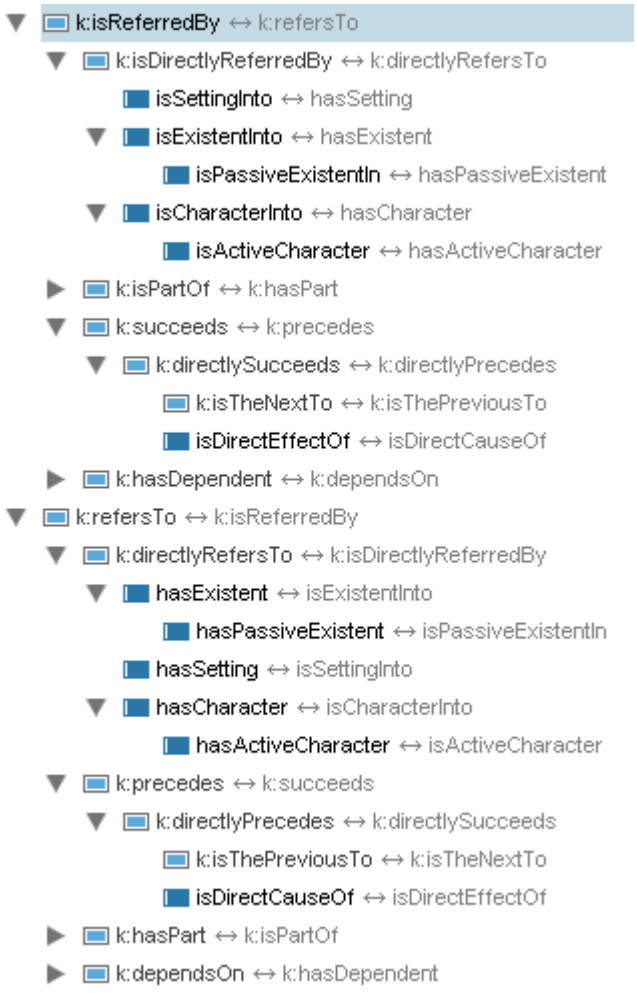

Figura 3.15: Jerarquía de roles del componente *Fabula* (1/2)

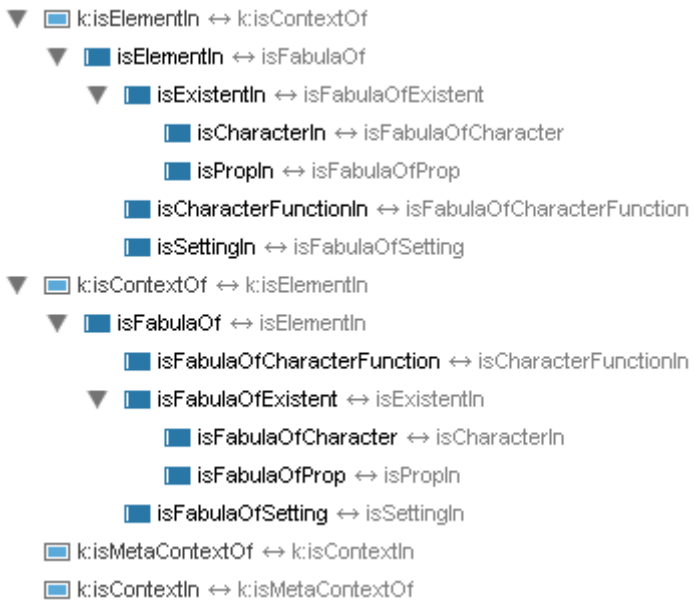

Figura 3.16: Jerarquía de roles del componente *Fabula* (2/2)

hasExistent y hasSetting relacionan una función de personaje con los existentes y los escenarios que participan en ella. El rol hasCharacter es más específico y sirve para relacionar una función de personaje con los personajes. Existen además dos roles adicionales, hasPassiveExistent y hasActiveCharacter que se utilizan para distinguir cuando un existente, habitualmente un accesorio, participa de forma pasiva en una función de personaje y cuando un personaje lo hace de forma activa.

Volviendo a los conceptos, los restantes son definidos y son ValuableFabula, ValuableCharacterFunction, ValuableCharacter, ValuableProp y ValuableSetting. ValuableFabula define una fábula valiosa según el criterio de este componente básico: debe contener un mínimo de tres funciones de personaje, alguna causa directa de otra función, alguna efecto directo de otra función y finalmente alguna de tipo no normalizado Main y tipo inferido ValuableCharacterFunction. ValuableCharacterFunction se define como aquella función de personaje con dos existentes, dos escenarios y al menos un personaje valioso de tipo ValuableCharacter. ValuableCharacter se define como un protagonista que participa en alguna función de tipo Main, o bien un personaje secundario o terciario que participa en una función de personaje valiosa. ValuableProp se define como un accesorio que participa en una función donde intervienen algún personaje primario como personaje activo. ValuableSetting se define como un escenario en el que ocurren más de tres funciones de personaje.

Con respecto a las operaciones sobre individuos normalizados, puede verse un resumen de la API de este componente en el Cuadro 3.5. El acceso, la modificación, la creación y la destrucción se realizan exclusivamente sobre individuos normalizados.

Acceso: A individuos Fabula, Setting, Character, Prop, CharacterFunction y todos sus individuos conectados, que serán devueltos según el tipo que les corresponda
Modificación: De individuos Setting, Character, Prop y especialmente CharacterFunction, siendo reordenadas las funciones de personaje respetando el orden causal
Creación: De individuos Setting, Character, Prop y CharacterFunction, pudiendo crear cadenas causales de estos últimos y habiendo métodos para incluir el personaje activo y el existente pasivo al crear una función de personaje
Destrucción: De individuos Setting, Character, Prop y CharacterFunction, ocurriendo un reorganización de las cadenas de funciones de personaje que se vean interrumpidas

Cuadro 3.5: Resumen de la API del componente *Fabula*

3.4.6. *Discourse*

El componente *Discourse* es un componente ontológico básico para el desarrollo de aplicaciones de narración automática utilizando *DLApplication*.

Este componente contiene la semántica declarativa y operacional del modelo de conocimiento propuesto para representar los discursos de las historias, la estructura de sus diferentes episodios. Los conceptos de este componente son *concretos* y modelan la narratología básica expuesta en el apartado 3.4.6. De todas formas se recomienda especializar este componente con un modelo de discurso más cercana al tipo de aplicación que se desee desarrollar para poder crear individuos más concretos. Este componente es considerado de *nivel medio*, ya que puede o no ser especializado por otro componente narrativo. Además sus conceptos tienen una jerarquía *poco fragmentada*. Este componente especializa al componente *Time*.

Una vista general de la jerarquía conceptual de este componente se muestra en la Figura 3.17.

Los conceptos normales de este componente son Discourse, Episode, Message y Referent. Discourse es la referencia al discurso de una historia concreta. Episode representa un episodio del discurso, cada una de las partes en que

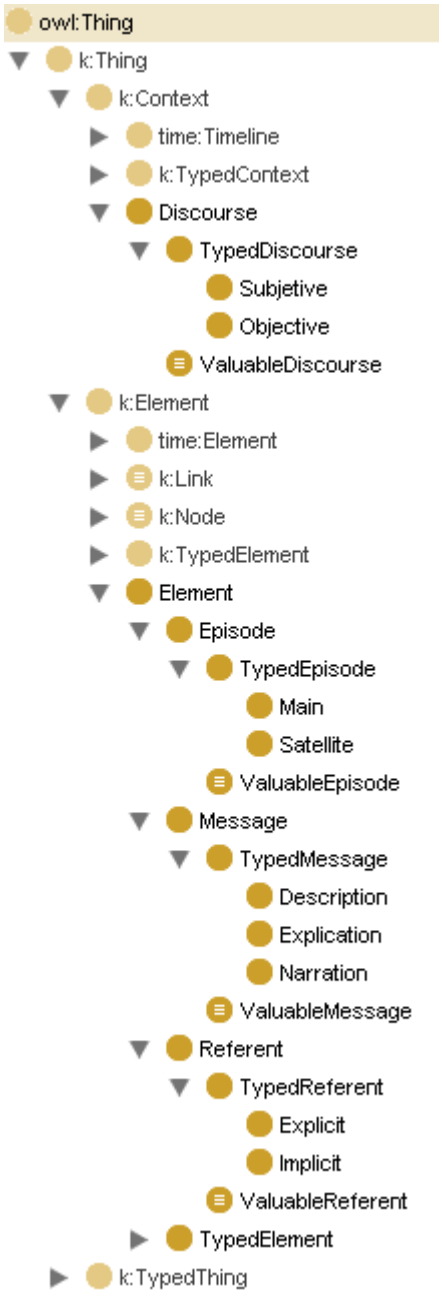

Figura 3.17: Jerarquía conceptual del componente *Discourse*

este se divide. Message representa un mensaje, el átomo del discurso y a su vez eslabón de la cadena temporal del mismo. El orden temporal que rige esta cadena es el orden en que será narrada la historia, que no tiene por qué coincidir con el orden temporal de los procesos de simulación en el mundo ficticio, siendo habitual aprovechar estas diferencias para crear distintos efectos narrativos que afectan al público, como las anacronías mencionadas en el apartado . Referent representa un referente que se menciona en algún mensaje del discurso, por ejemplo la forma en la que es posible referirse a un personaje o un accesorio de la fábula.

Los conceptos no normales son descendientes de TypedDiscourse, TypedEpisode, TypedMessage y TypedReferent. Los descendientes de TypedDiscourse son Subjective y Objective, y sirven para distinguir entre discursos subjetivos y objetivos. Los descendientes de TypedEpisode son Main y Satellite, dos tipos disjuntos que representan episodios principales o satélites, es decir, complementarios a los principales. Los descendiente de TypedMessage son Description, Explanation y Narration, tres tipos distintos de mensajes posibles que pueden aparecer en el discurso. Finalmente los descendientes de TypedReferent son Explicit e Implicit, según el referente sea de los que figurará finalmente de forma explícita en la presentación o no.

La jerarquía de roles de este componente se muestra en las Figuras 3.18 y 3.19.

Los roles de este componente son isDiscourseOfReferent, isDiscourseOfEpisode, isDiscourseOfMessage, hasReferent, hasMessage, hasSubEpisode, isThePreviousEpisodeTo, isThePreviousMessageTo y sus respectivas inversas. Los roles isDiscourseOfReferent, isDiscourseOfEpisode, isDiscourseOfMessage sirven para relacionar al discurso con sus referentes, episodios y mensajes, respectivamente. El rol hasReferent se utiliza para relacionar mensajes con los referentes que aparecen mencionados en ellos, al igual que el rol hasMessage se utiliza para relacionar episodios con los mensajes que aparecen en ellos. El rol hasSubEpisode y su inversa sirven para crear estructura jerárquicas en forma de árbol para los episodios, donde un episodio padre más general se descompone en una serie de episodios hijos más concretos, y el rol isThePreviousEpisodeTo y su inversa para conectarlos en forma de secuencia según el orden del discurso. El rol isThePreviousMessageTo y su inversa sirven para conectar mensajes de un episodio según el orden temporal de los mismos.

Volviendo a los conceptos, los restantes son definidos y son ValuableDiscourse, ValuableEpisode, ValuableMessage y ValuableReferent. ValuableDiscourse representa un discurso valioso según este componente básico, siendo definido como aquel discurso con al menos tres episodios y alguno de ellos de tipo inferido ValuableEpisode. ValuableEpisode se define como un episodio que posee al menos un subepisodio valioso o contiene un mensaje valioso.

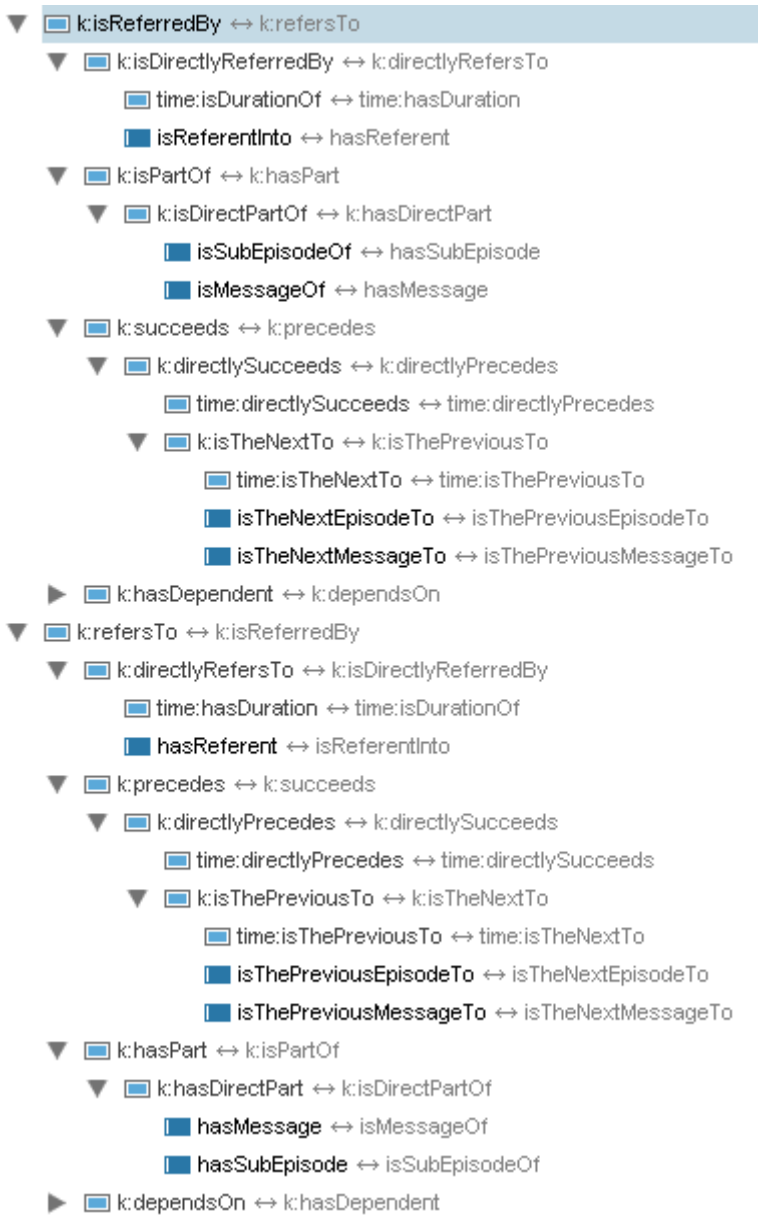

Figura 3.18: Jerarquía de roles del componente *Discourse* (1/2)

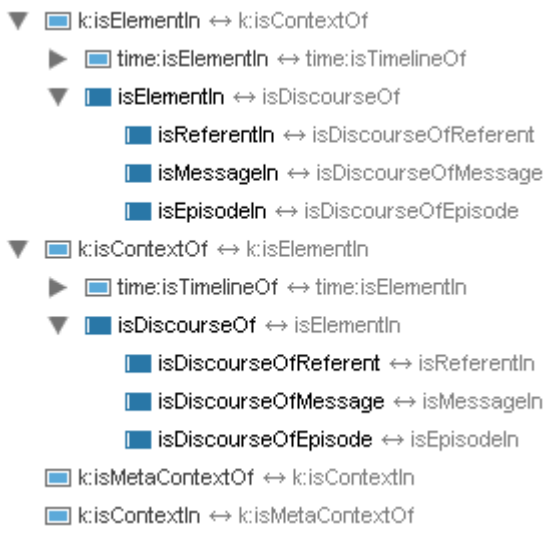

Figura 3.19: Jerarquía de roles del componente *Discourse* (2/2)

ValuableMessage se define como un mensaje con un máximo de tres referentes, teniendo algún referente valioso y alguno explícito, también. Finalmente ValuableReferent se define como un referente que participa en al menos dos mensajes, siendo al menos uno de ellos de tipo no normalizado Description.

Con respecto a las operaciones sobre individuos normalizados, puede verse un resumen de la API de este componente en el Cuadro 3.6. El acceso, la modificación, la creación y la destrucción se realizan exclusivamente sobre individuos normalizados.

Acceso: A individuos Discourse, Episode, Message y Referent **Modificación:** Especialmente de individuos Episode y Message, siendo reordenados respetando el orden temporal **Creación:** De individuos Discourse, Episode, Message y Referent, pudiendo crear cadenas temporales de los episodios y los mensajes **Destrucción:** De individuos Discourse y Episode, Message y Referent, ocurriendo una reorganización de las cadenas de episodios o mensajes que se vean interrumpidas y una destrucción en cascada de los subepisodios del episodio que es destruido

Cuadro 3.6: Resumen de la API del componente *Discourse*

3.4.7. *Narration*

El componente *Narration* es un componente ontológico básico para el desarrollo de aplicaciones de narración automática utilizando *DLApplication*.

Este componente contiene la semántica declarativa y operacional del modelo de conocimiento propuesto para representar la narración de una historia, combinación de fábula y discurso. Los conceptos de este componente son *concretos*, aunque lo más habitual es que cada aplicación especialice este componente con los elementos propios de las fábulas y discursos que van a aparecer en sus historias. Este componente es considerado de *nivel medio*, ya que puede o no ser especializado por otro componente. Además sus conceptos tienen una jerarquía de *fragmentación media*, dado que se espera que haya componentes que sólo especialicen el dominio de la fábula y otros el dominio del discurso. Este componente combina el componente *Fabula* con el componente *Discourse*.

Una vista general de la jerarquía conceptual de este componente se muestra en la Figura 3.20.

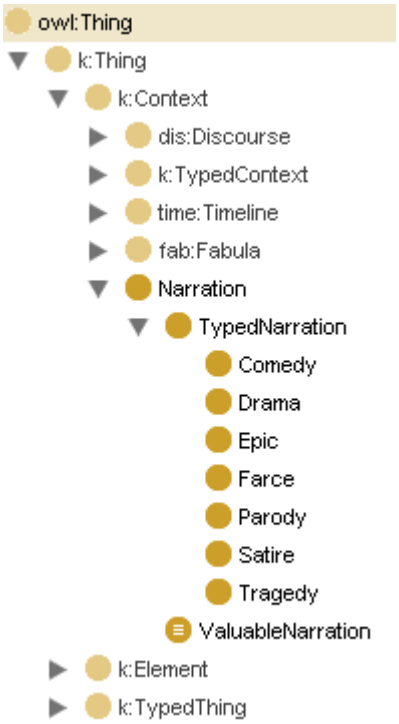

Figura 3.20: Jerarquía conceptual del componente *Narration*

El único conceptos normal del componente es Narration, la referencia al proceso mismo de la narración de una historia.

Los conceptos no normales son los descendientes de TypedNarration, géneros de las posibles historias: Comedy, Drama, Epic, Farce, Parody, Satire y Tragedy.

La jerarquía de roles de este componente se muestra en la Figura 3.21.

Los roles de este componente son isNarrationOfFabula, isNarrationOfDiscourse, hasCharacterFunction, hasExistent y sus correspondientes inversas. Los roles isNarrationOfFabula e isNarrationOfDiscourse sirven para conectar la narración con su fábula y discurso correspondientes. El rol hasCharacterFunction conecta un episodio con todas las funciones de personaje que se ponen de manifiesto dentro de él, mientras que el rol hasExistent conecta a cada referente con los existentes de la fábula a los que está haciendo referencia. Gracias a estos dos roles se conectan los elementos más importantes de la fábula con los del discurso, entretejiendo la estructura narrativa definitiva.

Volviendo a los conceptos, el único que resta por mencionar es definido y se llama ValuableNarration. Este concepto expresa el criterio de este componente básico a la hora de reconocer narraciones valiosas, concretamente aquellas que tienen tanto una fábula como un discurso valioso y que conectan al menos un episodio valioso con una función de personaje, y un referente valioso con un existente valioso.

Con respecto a las operaciones sobre individuos normalizados, puede verse un resumen de la API de este componente en el Cuadro 3.7. El acceso, la modificación, la creación y la destrucción se realizan exclusivamente sobre individuos normalizados.

Acceso: A individuos Narration
Modificación: De individuos Narration, para añadir fábula y discurso a una narración
Creación: De individuos Narration
Destrucción: De individuos Narration, destruyéndose en cascada la fábula y el discurso correspondientes

Cuadro 3.7: Resumen de la API del componente *Narration*

3.4.8. *AbstractStory*

El componente *AbstractStory* es un componente ontológico básico para el desarrollo de aplicaciones de narración automática utilizando *DLApplication*.

Este componente contiene la semántica declarativa y operacional del modelo de conocimiento propuesto para representar una historia abstracta, esto es, una historia que combina narración con simulación pero carece todavía de

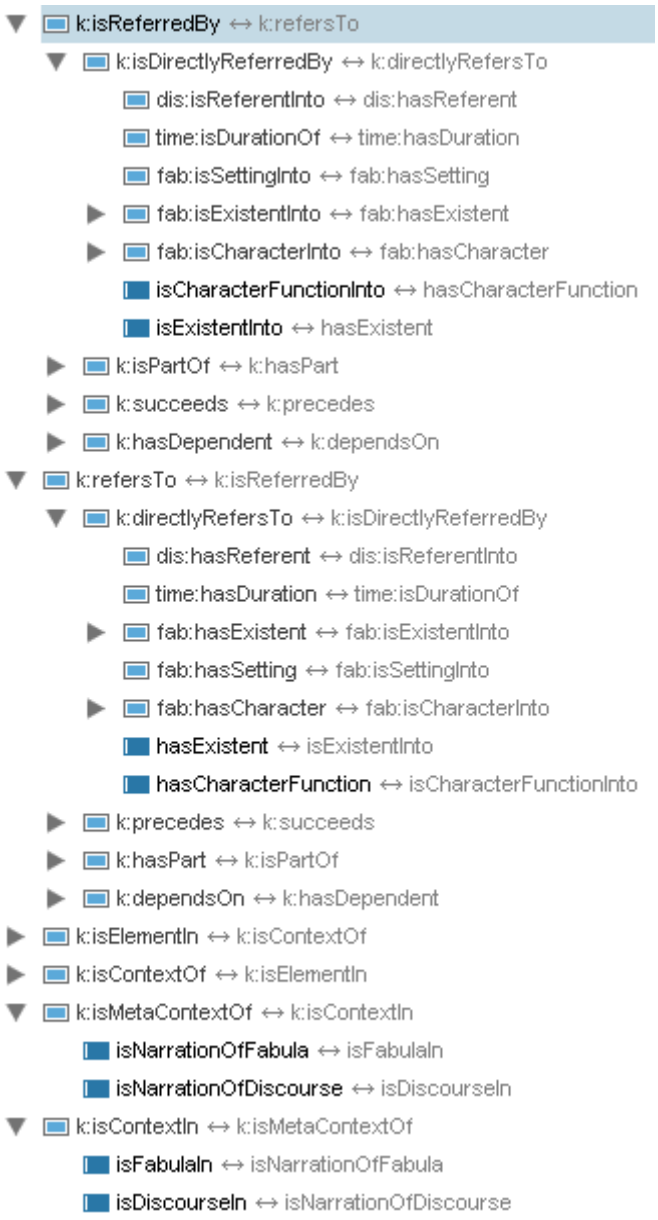

Figura 3.21: Jerarquía de roles del componente *Narration*

la forma definitiva con la que será presentada ante el público. Los conceptos de este componente son *concretos*, aunque lo más habitual es que cada aplicación especialice este componente con los elementos propios de las historias abstractas que genera. Este componente es considerado de *nivel medio*, ya que puede o no ser especializado por otro componente. Además sus conceptos tienen una jerarquía de *fragmentación media*, dado que se espera que haya componentes que sólo especialicen elementos relativos a la narración y otros sólo elementos relativos a la simulación. Este componente combina el componente *Narration* con el componente *Simulation*.

Una vista general de la jerarquía conceptual de este componente se muestra en la Figura 3.22.

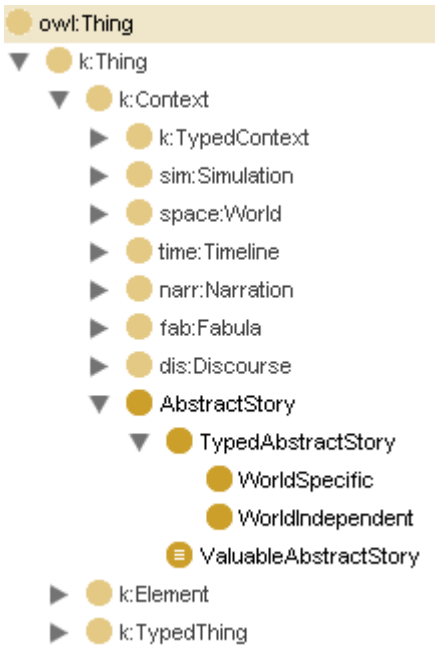

Figura 3.22: Jerarquía conceptual del componente *AbstractStory*

El único concepto normal de este componente es AbstractStory, la referencia a la historia abstracta, combinación de una narración y una simulación, y piedra angular del armazón propuesto, la raíz de cada historia generada.

Los conceptos no normales descienden de TypedAbstractStory y son dos conceptos disjuntos llamados WorldSpecific y WorldIndependent, que representan dos tipos de historias abstractas, según se considere que la narración depende de la simulación o funciona de manera independiente a esta.

La jerarquía de roles de este componente se muestra en las Figuras 3.23 y 3.24.

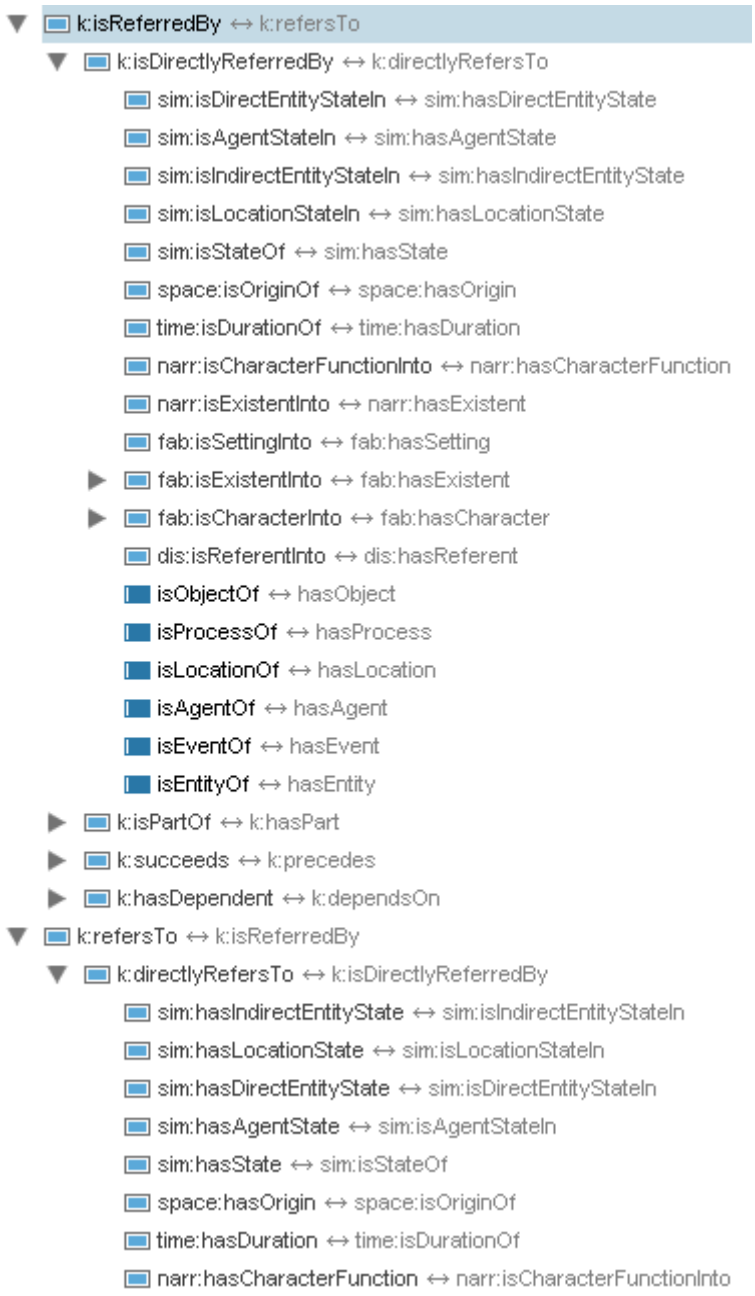

Figura 3.23: Jerarquía de roles del componente *AbstractStory* (1/2)

Figura 3.24: Jerarquía de roles del componente *AbstractStory* (2/2)

Los roles de este componente son isAbstractStoryOfNarration, isAbstractStoryOfSimulation, hasLocation, hasProcess, hasAgent, hasObject, hasEvent, hasEntity y sus respectivos roles inversos. Los roles isAbstractStoryOfNarration e isAbstractStoryOfSimulation sirven para conectar las historias abstractas con sus respectivas narraciones y simulaciones. El rol hasLocation conecta un escenario de la fábula con las localizaciones de la simulación que lo representan en el mundo ficticio. El rol hasProcess hace algo similar conectando una función de personaje con los procesos que la representan en la simulación. El rol hasAgent conecta un personaje con el agente o los agentes concretos que lo representan, y el rol hasObject hace lo mismo pero conectando accesorios con objetos de la simulación. Los roles hasEvent y hasEntity tienen una funcionalidad análoga, ya que sirven para conectar mensajes y referentes del discurso con los eventos y entidades correspondientes de la simulación. Todos estos roles juntos ayudan a completar el grafo de una historia abstracta, el esquema más completo posible de lo que se puede generar mediante este armazón de narración automática.

Volviendo a los conceptos, el único que falta por comentar es definido y se llama ValuableAbstractStory, el tipo inferido de las historias abstractas consideradas valiosas en el sistema, definido como la historia que tiene narración valiosa, simulación valiosa y que además conecta una función de personaje valiosa con al menos un proceso valioso, y un referente valioso con al menos una entidad valiosa.

Con respecto a las operaciones sobre individuos normalizados, puede verse un resumen de la API de este componente en el Cuadro 3.8. El acceso, la modificación, la creación y la destrucción se realizan exclusivamente sobre individuos normalizados.

Acceso: A individuos AbstractStory
Modificación: De individuos AbstractStory, para añadir narración y simulación a una historia abstracta
Creación: De individuos AbstractStory
Destrucción: De individuos AbstractStory, destruyéndose en cascada la correspondiente narración y la simulación

Cuadro 3.8: Resumen de la API del componente *AbstractStory*

3.5. Funcionamiento del armazón

En las APIs de todos los componentes presentados existe además un método de *generación* por cada tipo normalizado, que es el responsable de gene-

rar nuevos individuos normalizados. Los componentes básicos proporcionan una implementación muy básica de este método de generación, pensada para ser extendida o en muchos casos totalmente sustituida por implementaciones más específicas de la aplicación que se desea construir.

El armazón por tanto no está limitado a un algoritmo concreto de generación de historias, sino que proporciona una infraestructura distribuida donde la generación de un individuo normalizado de tipo AbstractStory, por ejemplo, desencadena la generación en cadena de todos los individuos relacionados con este hasta llegar a formar por completo la estructura de una historia abstracta. Esto permite que en cualquier punto del proceso, en la generación de cualquier individuo de tipo normalizado, pueda existir un algoritmo específico de dominio diferente.

Lo que el armazón, a través de la implementación de sus componentes básicos, ofrece es un algoritmo general que puede describirse mediante este pseudocódigo:

```
procedimiento generar(Normal : x) → booleano
   hacer
      si no generarTipos(x) entonces
         devolver falso
      si no generarRelaciones(x) entonces
         devolver falso
      generarTipos'(x)
      generarRoles'(x)
   mientras repetirEvaluacion(valor(x), novedad(x))
   devolver cierto
fin procedimiento
```

siendo:

- Normal el tipo normalizado de x
- x el individuo, previamente creado, que se va a normalizar
- $generarTipos(x)$ la generación de tipos no normalizados necesarios para x
- $generarRelaciones(x)$ la generación de relaciones necesarias entre x y otros individuos
- $generarTipos'(x)$ la generación de tipos no normalizados opcionales para x

- *generarRelaciones'*(x) la generación de relaciones opcionales entre x y otros individuos
- *valor*(x) la métrica de valor para el tipo Normal
- *novedad*(x) la métrica de novedad para el tipo Normal
- *repetirEvaluacion*(v, n) la función que decide si hay que repetir la generación debido a una evaluación negativa o no

La función *generarTipos*(x) se describe a su vez mediante este pseudocódigo:

```
procedimiento generarTipos(Normal : x) → booleano
    contador ← 1
    hacer
        T ← tipos(x, contador)
        para t ← 1 hasta |T| hacer
            ponerTipo(T_t, x)
        fin para
        contador ← contador + 1
    mientras repetirTipos(x, contador)
    si tiposIncompletos(x) entonces
        devolver falso
    fin si
fin procedimiento
```

siendo:

- *tipos*$(x, contador)$ un posible conjunto de tipos no normalizados necesarios para completar un individuo x
- *ponerTipo*(t, x) la función que añade el tipo t al individuo x
- *repetirTipos*$(x, contador)$ la función que decide si hay que repetir esta generación o no
- *tiposIncompletos*(x) la función que decide si esta generación ha fracasado o no

La función *generarRelaciones*(x) se describe a su vez mediante este pseudocódigo:

```
procedimiento generarRelaciones(Normal : x) → booleano
  contador ← 1
  hacer
    R ← roles(x, contador)
    para r ← 1 hasta |R| hacer
      para i ← 1 hasta relacionados(x, R_r, contador) hacer
        y_{r,i} ← crear(tipoRelacionado(x, R_r, contador))
        relacionar(x, R_r, y_{r,i})
      fin para
    fin para
    fallos ← ∅
    para r ← 1 hasta |R| hacer
      para i ← 1 hasta relacionados(x, R_r, contador) hacer
        si no generar(y_{r,i}) entonces
          fallos ← fallos ∪ y_{r,i}
        fin si
      fin para
    fin para
    contador ← contador + 1
  mientras repetirRelaciones(x, contador, fallos)
  si relacionesIncompletas(x) entonces
    devolver falso
  fin si
fin procedimiento
```

siendo:

- $roles(x, contador)$ un posible conjunto de roles no normalizados necesarios para completar un individuo x
- $relacionados(x, r, contador)$ el número de individuos que deben estar necesariamente relacionados con x a través del rol r para completarlo
- $tipoRelacionado(x, r, contador)$ el tipo que corresponde al individuo contador-ésimo necesariamente relacionado con x a través del rol r
- $crear(Y)$ la función que crea a un individuo de tipo normalizado Y
- $relacionar(x, r, y)$ la función que relaciona al individuo x con el individuo y a través del rol r
- $repetirRelaciones(x, contador, fallos)$ la función que decide si hay que repetir esta generación o no

- $relacionesIncompletas(x)$ la función que decide si esta generación ha fracasado o no

La función $generarTipos'(x)$ se describe a su vez mediante este pseudocódigo:

```
procedimiento generarTipos'(Normal : x)
   contador ← 1
   hacer
      T'← tipos'(x, contador)
      para t ← 1 hasta |T'| hacer
         ponerTipo(T'_t, x)
      fin para
      contador ← contador + 1
   mientras repetirTipos'(x, contador)
fin procedimiento
```

siendo:

- $tipos'(x, contador)$ un posible conjunto de tipos no normalizados opcionales para enriquecer un individuo x
- $repetirTipos'(x, contador)$ la función que decide si hay que repetir esta generación o no

La función $generarRelaciones'(x)$ se describe a su vez mediante este pseudocódigo:

```
procedimiento generarRelaciones'(Normal : x)
   contador ← 1
   hacer
      R'← roles'(x, contador)
      para r ← 1 hasta |R'| hacer
         para i ← 1 hasta relacionados'(x, R'_r, contador) hacer
            relacionar(x, R'_r, z_r,i)
         fin para
      fin para
      contador ← contador + 1
   mientras repetirRelaciones'(x, contador)
   devolver cierto
fin procedimiento
```

siendo:

- $roles'(x, contador)$ un posible conjunto de roles no normalizados opcionales para enriquecer un individuo x

- $relacionados'(x, r, contador)$ el número de individuos que deben estar opcionalmente relacionados con x a través del rol r

- $repetirRelaciones'(x, contador)$ la función que decide si hay que repetir esta generación o no

En el siguiente apartado se aportan más detalles sobre las métricas de evaluación del valor y la novedad de una historia.

3.6. Evaluación del armazón

Los componentes básicos que acaban de ser descritos incluyen una batería de pruebas automáticas de acuerdo al orden sugerido para probar las aplicaciones desarrolladas con este armazón.

En primer lugar se deben ejecutar las pruebas del núcleo software y del componente raíz *Knowledge*. En segundo lugar se deben ejecutar las pruebas de los componentes básicos, comenzando desde los más simples como *Space*, *Fabula* y *Time* hasta los más complejos como *Simulation*, *Discourse*, *Narration* y *AbstractStory*. En tercer lugar deberán ser ejecutadas las baterías de pruebas que tengan aquellos componentes específicos que sustituyen, especializan o combinan a los básicos, siempre en orden de complejidad creciente. Finalmente se realizarán pruebas de componentes de metadominios, siempre conectándolos con otros componentes de dominio en orden de complejidad creciente.

Como complemento a esta serie de pruebas automáticas, el desarrollador de una aplicación puede incorporar más métodos para comprobar a fondo el comportamiento de operaciones o combinaciones de operaciones concretas que presentan un mayor riesgo de error en el contexto de su aplicación.

El propósito de este armazón es desarrollar aplicaciones de narración automática, y a su vez el propósito de estas aplicaciones es generar historias coherentes que resulten suficientemente valiosas y novedosas según unos criterios preestablecidos por el desarrollador. Es por ello, que además de asegurar el buen funcionamiento de los distintos componentes realizando las baterías de pruebas, la forma más eficaz de evaluar el funcionamiento del armazón es otorgar un método de evaluación integrado para sus aplicaciones, de manera que puedan medirse según las historias que generan, ya que éstas son en realidad el artefacto más accesibles de todo este complejo sistema.

De hecho, la evaluación de estas historias forma parte de la metodología de desarrollo de aplicaciones del armazón propuesto en esta investigación, sirviendo siempre como realimentación de dicho desarrollo. Distinguimos dos métodos diferentes pero complementarios de evaluación: la *evaluación empírica* y la *evaluación formal*.

La evaluación empírica se basa en medir la satisfacción de los usuarios finales con respecto a las historias generadas, tras haber tenido acceso a ellas. Este tipo de evaluación resulta especialmente adecuado para obtener datos cualitativos y subjetivos en relación al desarrollador o miembro del público que lo evalúa. Aunque es posible pedir directamente a dichos usuarios que otorguen un valor numérico representativo del grado de *satisfacción* que les produce cada historia generada, resulta más útil descomponer la satisfacción global en varios factores.

En cualquier caso el medio habitual para recopilar esta información empírica son los *cuestionarios de evaluación*. Estos cuestionarios los rellena el usuario inmediatamente después de haberle sido presentadas las historias a evaluar. Al usuario se le piden inicialmente datos generales como su sexo, edad, nivel de estudios, profesión, etc. Tras las preguntas sobre los factores de satisfacción se sitúa un campo de texto libre para añadir cualquier comentario, crítica o sugerencia que no se pueda reflejar en otra parte. Finalmente se solicitan, de manera opcional, datos personales o de contacto para poder enviar los resultados al usuario al término de esta investigación.

La evaluación formal se basa en calcular una previsión de la satisfacción de los usuarios finales de acuerdo a una serie de métricas numéricas establecidas de antemano y normalmente basadas en criterios heurísticos. Al ser datos fácilmente calculables, es posible integrar su cálculo en la propia aplicación, como parte de un sistema de aprendizaje automático o envío de realimentación para los desarrolladores. Este tipo de evaluación resulta especialmente adecuado para obtener datos cuantitativos de forma objetiva.

Los factores principales que se consideran en esta propuesta son la *coherencia*, el *valor*, la *novedad* y la *calidad de presentación*. Para la evaluación empírica todos estos factores son medidos en escalas intuitivas y uniformemente distribuidas, utilizando los números enteros comprendidos entre 1 y 5 para después poder aplicar medidas estadísticas. Para la evaluación formal se utiliza una escala diferente, números reales entre 0 y 1, ambos incluidos; y se emplean métricas concretas que calculan de forma recursiva la valoración global de cada factor, acumulando las valoraciones locales que se obtienen en la evaluación de cada componente y dominio de conocimiento correspondiente. De este modo, a cada valoración local se le puede asignar un peso distinto y tener de esa forma mayor flexibilidad a la hora de ajusta la evaluación concreta de una aplicación.

3.6.1. Coherencia

La coherencia mide el grado de encadenamiento lógico que existe entre los elementos de una historia. Además de la coherencia global, la coherencia se mide en cada dominio de conocimiento, siendo especialmente relevante la coherencia de simulación, dada la predisposición que tiene el ser humano para entender que cualquier mundo, aunque sea ficticio, debe ser consistente de acuerdo a una cierta lógica, en términos de relaciones espaciales entre las entidades y las localizaciones del mundo ficticio y las relaciones temporales entre las secuencias de eventos de dicha simulación. La coherencia en el dominio de la fábula es también importante, dado que el público es capaz de identificar con bastante claridad las relaciones de causalidad que existen entre las distintas funciones de personaje.

En términos formales, y dentro del contexto del armazón propuesto, la coherencia en un determinado dominio de conocimiento será un valor booleano que indica si cada uno de los individuos de ese dominio son o no coherentes, entendiendo esto como la unión de dos características: *consistencia* lógica bajo la premisa de mundo abierto y *completitud* bajo la premisa de mundo cerrado.

El armazón propuesto garantiza que en ningún momento puede hacerse efectiva una modificación que haga desaparecer la consistencia lógica del modelo interno que está utilizando la aplicación. Sin embargo, como se deduce de su definición, en este trabajo la coherencia es un requisito más fuerte que la mera consistencia, que se mantiene automáticamente, sino que además es necesario que exista completitud, usando los métodos de generación apropiados para los individuos normalizados del API de cada componente que se encargan de crear y relacionar todos los individuos necesarios para que la estructura entera resulte completa.

En el dominio de la fábula, por ejemplo, la causalidad es la relación de coherencia por excelencia, y en el caso de otros dominios, las relaciones físicas y espacio-temporales son también muy importantes.

3.6.2. Valor

El valor en el contexto de este armazón es el factor que mide el grado de interés que la historia posee de acuerdo al criterio de sus autores, ya sea educativo, estético, lúdico o de otro tipo. En el caso de una historia generada automáticamente donde el usuario no ha impuesto ningún criterio de valor, consideramos que este factor mide el grado de interés general que la historia despierta según su propio criterio subjetivo.

En términos formales el valor de una historia es el valor del individuo

de tipo AbstractStory que lo representa. En general el valor de un individuo normalizado se calcula mediante una función $valor(x)$ específica de cada concepto normal de cada componente, siendo el único requisito que esta función devuelva un valor real entre 0 y 1, representando el mínimo y el máximo valor posible que puede representar un individuo x.

Todos los componentes básicos que ofrece este armazón implementan la misma función $valor(x)$ para los individuos x de todos los tipos normalizados X. Esta función es bastante sencilla, y se define recursivamente de la siguiente manera:

$$valor(x) = v(x) \cdot p_0 + \sum_{i=1}^{roles(x)} \left(\frac{\sum_{j=1}^{otros(x,R(x)_i)} valor(I(x, R(x)_i)_j)}{otros(x, R(x)_i)} \cdot p_i \right)$$

siendo:

- $v(x)$ el valor booleano de un individuo
- $R(x)$ los roles usados por un individuo
- $roles(x)$ el total de roles usados por un individuo
- $I(x, r)$ los individuos relacionados con otro mediante un rol
- $otros(x, r)$ el total de individuos relacionados con otro mediante un rol
- p_0 y p_i los pesos asignados a cada sumando

Esta función es la suma de dos partes fundamentales: una parte *lógica*, dependiente del resultado de aplicar la función v a x, y otra parte *numérica*, dependiente del resultado de aplicar la función *valor* correspondiente a todos los individuos relacionados con el individuo x.

En relación a la parte lógica, la función de asignación de un valor booleano de un individuo x se define así:

$$v(x) = \begin{cases} 1 & x \in \mathsf{Valuable} \\ 0 & x \notin \mathsf{Valuable} \end{cases}$$

siendo Valuable el concepto definido ancestro de todos los conceptos definidos que reconocen individuos "valiosos" en un determinado dominio.

En relación a la parte numérica, el conjunto de roles usados $R(x)$ y la función total de roles usados $roles(x)$ sirven para identificar todos los roles r

que relacionan al individuo x con otro individuo de la base de conocimiento del modelo, y para contabilizar el número total de ellos, respectivamente. Sus definiciones formales son:

$$R(x) = \{r | \exists y / < x, r, y > \in KB\}$$

$$roles(x) = |R(x)|$$

Siguiendo con la parte numérica, el conjunto de individuos relacionados $I(x, r)$ y la función total de individuos relacionados $otros(x, r)$ sirven para identificar todos los individuos relacionados con x a través del rol r en la base de conocimiento del modelo, y para contabilizar el número total de ellos, respectivamente. Sus definiciones formales son:

$$I(x, r) = \{y | < x, r, y > \in KB\}$$

$$otros(x, r) = |I(x, r)|$$

Por último, es importante aclarar que los pesos p_i de los distintos sumandos que hay en cada parte son establecidos en el fichero de configuración del dominio correspondiente a cada componente ontológico, debiendo respetar siempre estas condiciones:

$$p_i \geq 0 \quad \forall i, i \in [0..roles(x)]$$

$$\sum_{i=0}^{roles(x)} p_i = 1$$

3.6.3. Novedad

La novedad en el contexto de este armazón es el factor que mide las diferencias que existen entre una historia y todas las demás que se encuentran en la base de conocimiento en el momento justo de su generación. Esta medida se corresponde, en términos de creatividad computacional, con la c-creatividad de la aplicación. Para intentar medir algo equivalente a la h-creatividad de la aplicación, habría que comparar la historia generada con todas las existentes, dentro o fuera del sistema, lo cual es completamente inviable. Una aproximación posible a esta medida sería comparar la historia generada con un corpus representativo de historias similares, incluyendo en este corpus numerosas historias que no estén presentes en la base de conocimiento. Esto es

algo que ocurre de forma implícita al pedir al usuario que evalúe la novedad de una historia, ya que el corpus de comparación lo forman todas aquellas historias que el usuario conoce y considera similares a la que le está siendo presentada.

En términos formales la novedad de una historia es la novedad del individuo de tipo AbstractStory que lo representa. En general la novedad de un individuo normalizado se calcula mediante una función $novedad(x)$ específica de cada concepto normal de cada componente, siendo el único requisito que esta función devuelva un valor real entre 0 y 1, representando la mínima y la máxima novedad posible que puede representar un individuo x.

Todos los componentes básicos que ofrece este armazón implementan la misma función $novedad(x)$ para todos los individuos x de todos los tipos normalizados X. Esta función es muy intuitiva, y se define de la siguiente manera:

$$novedad(x) = 1 - max\{similitud(x, y) | \forall y, (y \in \mathsf{X} \wedge y \neq x)\}$$

siendo:

- X el tipo normalizado de x
- $similitud(x, y)$ la similitud entre dos individuos de tipo X

La medida de similitud entre dos individuos normalizados también se calcula mediante una función específica de cada concepto normal, siendo el único requisito que esta función devuelva un valor real entre 0 y 1, representando la mínima y la máxima similitud posible que puede existir entre dos individuos x e y. Todos los componentes básicos que ofrece este armazón implementan la misma función $similitud(x, y)$ para los individuos de todos los tipos normalizados. Esta función ignora cualquier diferencia debida a que y tenga más tipos, roles o individuos relacionados que x y se centra en comprobar si hay algo en x que no se encuentre en y, definiéndose recursivamente de la siguiente manera:

$$\begin{aligned} similitud(x, y) = &\left(1 - \frac{soloTipos(x, y)}{tipos(x)}\right) \cdot p_T + \left(1 - \frac{soloRoles(x, y)}{roles(x)}\right) \cdot p_R \\ &+ \sum_{i=1}^{roles(x,y)} \Bigg(\left(1 - \frac{|otros(x, R(x)_i) - otros(y, R(x)_i)|}{otros(x, R(x)_i) + otros(y, R(x)_i)}\right) \cdot p_{i_1} \\ &+ similitud(x, y, R(x)_i) \cdot p_{i_2}\Bigg) \end{aligned}$$

siendo:

- $tipos(x)$ el total de tipos que tiene un individuo
- $soloTipos(x, y)$ el total de tipos de tiene un individuo y no tiene otro
- $R(x)$ los roles usados por un individuo
- $roles(x)$ el total de roles usados por un individuo
- $roles(x, y)$ el total de roles usados por dos individuos
- $soloRoles(x, y)$ el total de roles usados por un individuo y no por otro
- $I(x, r)$ los individuos relacionados con otro mediante un rol
- $otros(x, r)$ el total de individuos relacionados con otro mediante un rol
- $similitud(x, y, r)$ similitud entre los individuos relacionados con dos individuos distintos mediante un mismo rol
- p_T, p_R y p_i los pesos asignados a cada sumando

Esta función es la suma de tres partes fundamentales: una parte relativa a los tipos de los individuos, otra parte relativa a los roles utilizados y una última parte relativa a los tipos de los individuos relacionados con los individuos x e y que se están comparando.

En relación a los tipos de los individuos, la función $tipos(x)$ sirve para contabilizar todos los tipos que tiene el individuo x, y la función $soloTipos(x, y)$ sirve para contabilizar todos los tipos que tiene el individuo x pero no tiene el individuo y. Sus definiciones formales son:

$$tipos(x) = |\{T|x \in T\}|$$

$$soloTipos(x, y) = |\{T|x \in T \wedge y \notin T\}|$$

En relación a los roles utilizados, las funciones $R(x)$ y $roles(x)$ son las mismas que las vistas en el apartado de evaluación del valor de una historia. La función $roles(x, y)$ sirve para contabilizar todos los roles que relacionan tanto al individuo x como al individuo y con otros individuos. Las función $soloRoles(x, y)$ sirve para contabilizar todos los roles que relacionan al individuo x con otro individuo pero no relacionan al individuo y con ningún individuo. Las definiciones formales de $roles(x, y)$ y $soloRoles(x, y)$ son semejantes:

$$roles(x, y) = |\{r|\exists z/ < x, r, z > \in KB \wedge \exists z/ < y, r, z > \in KB\}|$$

$$soloRoles(x,y) = |\{r|\exists z/ < x,r,z > \in KB \wedge \nexists z/ < y,r,z > \in KB\}|$$

En relación a los individuos relacionados, las funciones $I(x,r)$ y $otros(x,r)$ son las mismas que las vistas en el apartado de evaluación del valor de una historia. La función $similitud(x,y,r)$ es una función auxiliar que sirve para comparar el conjunto de individuos relacionados con x a través del rol r con el conjunto de todos los individuos relacionados con y a través del mismo rol, definiéndose así:

$$similitud(x,y,r) = \frac{\sum_{i=1}^{otros(x,R(x)_i)} max\{similitud(I(x,r)_i,z)|\forall z, z \in I(y,r)\}}{otros(x,r)}$$

Por último, es importante aclarar que los pesos p_T, p_R y p_i de los distintos sumandos que hay en cada parte son establecidos en el fichero de configuración del dominio correspondiente a cada componente ontológico, debiendo respetar siempre estas condiciones:

$$p_T \geq 0 \quad p_R \geq 0$$

$$p_{i_1} \geq 0 \wedge p_{i_2} \geq 0 \quad \forall i, i \in [1..roles(x)]$$

$$p_T + p_R + \sum_{i=1}^{roles(x)} (p_{i_1} + p_{i_2}) = 1$$

3.6.4. Calidad de presentación

La calidad de presentación es el factor que mide la satisfacción del público con el uso del medio narrativo empleado para presentar la historia. Este factor es muy importante, aunque dependiente del medio narrativo y la tecnología de presentación utilizados. En el caso de las historias presentadas en forma de texto, es la calidad lingüística o de uso del lenguaje, cuya evaluación puede dividirse a su vez en subfactores o dejarse al criterio subjetivo del usuario final de la aplicación.

La definición de métricas formales para evaluar la calidad de la presentación en un medio concreto excede los objetivos propuestos para esta investigación. En general cualquier factor de satisfacción que dependa directamente de la forma en que es presentado al público no se tendrá en consideración en la evaluación formal.

Capítulo 4

Validación del armazón

En teoría, no hay diferencia entre teoría y práctica.
Pero en la práctica, la hay.
Jan L.A. van de Snepscheut

En este capítulo se presenta la validación del armazón presentado con detalle en el capítulo 3. Esta validación se lleva a cabo realizando una instancia del armazón, es decir, desarrollando una aplicación de narración automática que sirve para probar la funcionalidad del mismo. Esta aplicación pone en práctica las ideas de este trabajo, las implementa e ilustra de una forma concisa, permitiendo evaluarlas y probar así la viabilidad práctica de esta tesis doctoral.

El nombre de esta aplicación es *ProtoPropp* [PGDAH], y se trata de una aplicación generadora de historias cortas en lenguaje natural muy sencillo. El dominio de aplicación es el de los cuentos maravillosos del folklore ruso, centrándose principalmente en el correcto modelado de las fábulas vistas según la morfología narrativa proppiana expuesta en el apartado 3.4.5.

Esta aplicación será un generador simple de historias "básicas", llamadas así porque su presentación no será compleja, sino una conversión muy directa de las historias abstractas generadas a texto. Además ilustrará la posibilidad de ampliar el número de componentes básicos del armazón añadiendo componentes específicos de la aplicación, para especializar el contenido de las fábulas y la simulación de los cuentos, además de para añadir un metadominio de apoyo para el razonamiento. Los componentes específicos *KICBR*, *Propp* y *Tale* son descritos en el apartado 4.1.

Utilizando estos componentes y una base de casos de fábulas de cuentos populares rusos, la aplicación es capaz de generar cuentos nuevos realizando una recuperación y una adaptación basada en la similitud y las relaciones ontológicas que se dan entre dichas funciones.

La aplicación realiza una adaptación de varios cuentos existentes en una base de casos para crear un cuento nuevo. Esta adaptación siempre se realiza mediante la sustitución de unas funciones de personaje proppianas por otras del mismo tipo normalizado o de un tipo ancestro o descendiente, respetando unas las restricciones que vienen incluidas en las declaraciones de las funciones de personaje para que la estructura del cuento no pierda coherencia.

4.1. Componentes específicos de la aplicación

En la Figura 4.1 se muestra la jerarquía de componentes específicos desarrollados para *ProtoPropp*. Como ya se mencionó al hablar de los componentes básicos, se trata de una polijerarquía que admite herencia múltiple, a través tanto de las ontologías OWL DL como de los interfaces *Java*. Los componentes de color gris son los componentes básicos descritos en el capítulo 3. Los componentes de color blanco, incluyendo la base de conocimiento, son implementaciones desarrolladas específicamente para esta aplicación y que forman parte de los resultados de investigación.

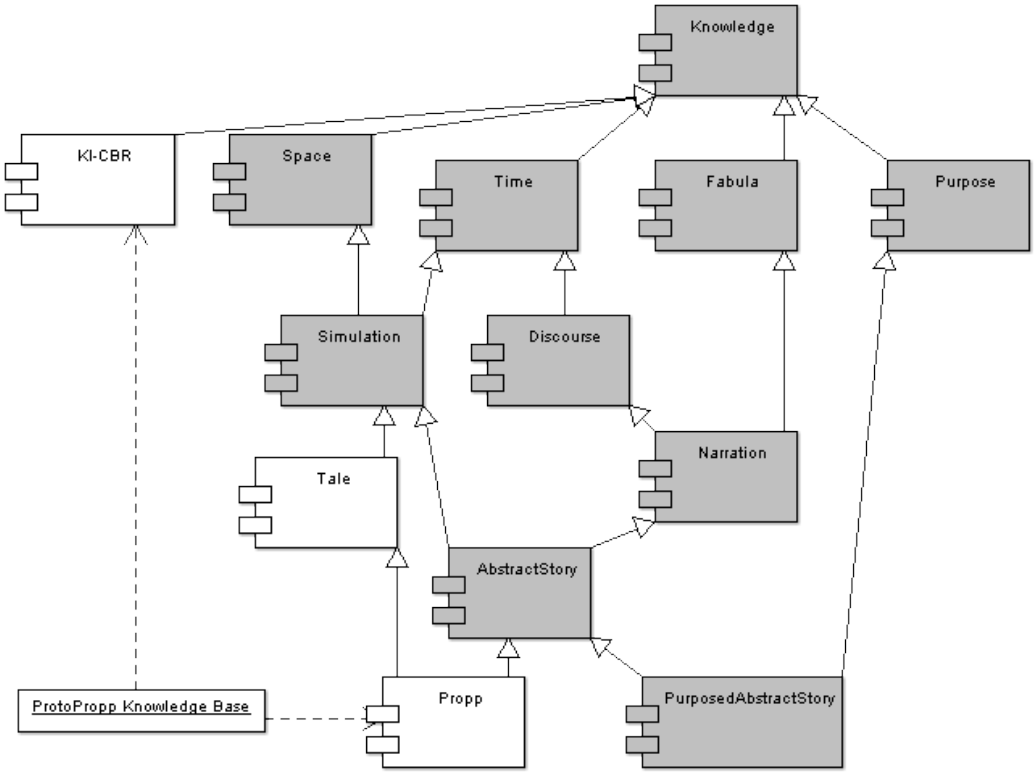

Figura 4.1: Jerarquía de componentes específicos de *ProtoPropp*

4.1.1. *Tale*

El componente *Tale* es un componente ontológico específico para el desarrollo de esta aplicación.

Este componente contiene la semántica declarativa y operacional del modelo de conocimiento propuesto para representar la simulación del mundo ficticio de un cuento maravilloso, con los procesos, agentes y objetos que suelen aparecer en este tipo de historias. Para los procesos se toman algunas ideas de la Teoría de la Dependencia Conceptual expuesta en el apartado 2.3.1. Los conceptos de este componente son *concretos*. Este componente es considerado de *nivel bajo*, ya que no se espera que sea especializado por otro componente. Además sus conceptos tienen una jerarquía de *fragmentación media*, heredada del dominio de la simulación, que modela muchas facetas de la realidad distintas. Este componente especializa al componente *Simulation*.

Una vista general de la jerarquía conceptual de este componente se muestra en las Figuras 4.2, 4.3 y 4.4.

Los conceptos normales de este componente especializan conceptos importantes de componentes básicos como *Simulation* o *Space*. Estos conceptos padres son Agent, Object, Location y Process. Bajo Agent aparecen conceptos disjuntos como HumanBeing (visto en esta ocasión como agente y no sólo como entidad física), Animal, Plant, Artifact y Spirit, que incluso se ve dividido en más subtipos. Bajo Object aparecen conceptos como Wear o Weapon, entre otros. Bajo Location aparecen conceptos como Dwelling, Village y OpenLocation, con más subtipos a su vez. Finalmente como subtipos de Process aparecen conceptos concretos inspirados en las primitivas de la Teoría de la Dependencia Conceptual, a los que se han añadido algunos otros todavía más específicos, como Love, Feel, Marry, Devour o Dance que resultan de utilidad para modelar este dominio. Aunque no se despliegan todos por razones de espacio y claridad en las explicaciones, los procesos concretos más importantes son: Attack, Attend, Change, Create, Destroy, Drop, Equip, Expel, Give, Go, Inform, Ingest, Move, Propel, Speak y Take.

Los conceptos no normales son descendientes de TypedHumanBeing, principalmente, pero también de TypedObject y TypedLocation. Los descendientes de TypedHumanBeing son numerosos tipos, la mayor parte no disjuntos, que actúan como "adjetivos" de los agentes de la simulación. Aunque no se despliegan completamente en la figura, algunos de ellos son Blonde, Coward, Elegant y Envious. Los descendientes de TypedObject son Metallic y Wooden, y los descendientes de TypedLocation son Wonderful, Enchanted y Scary.

Este componente no añade nuevos roles, ya que se centra sólo en enriquecer el vocabulario conceptual del modelo.

Volviendo a los conceptos, los conceptos definidos de este componente

Figura 4.2: Jerarquía conceptual del componente *Tale* (1/3)

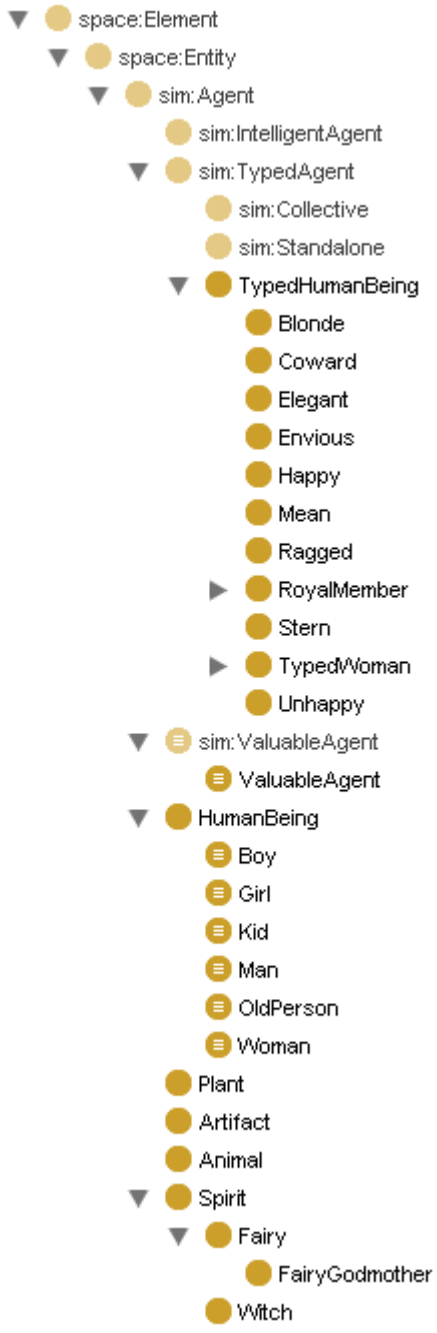

Figura 4.3: Jerarquía conceptual del componente *Tale* (2/3)

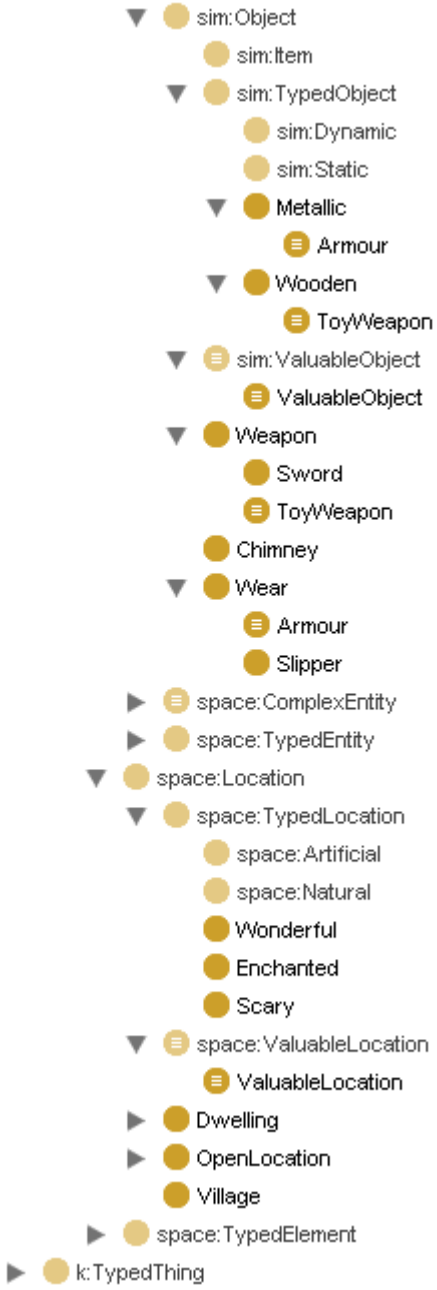

Figura 4.4: Jerarquía conceptual del componente *Tale* (3/3)

son Boy, Girl, Kid, Man, Woman, Armour, ToyWeapon, OldPerson, ValuableSimulation, ValuableProcess, ValuableAgent, ValuableObject y ValuableLocation. Boy, Girl, Kid, Man, Woman y OldPerson son tipos inferidos auxiliares que sirven para reconocer agentes característicos de los cuentos, según su sexo y su edad. Algo similar ocurre con Armour y ToyWeapon que reconocen las prendas de vestir metálicas y las armas de madera como armaduras y armas de juguete, respectivamente. ValuableSimulation representa lo que en este componente se entiende por una simulación valiosa, siendo su definición formal la de especializar a ValuableSimulation de *Simulation* y contener un mundo donde existe al menos un agente de tipo ValuableAgent de este componente (no de *Simulation*), y una línea temporal donde existe algún proceso de tipo ValuableProcess también de este componente. ValuableProcess especializa el concepto ValuableProcess de *Simulation*, definiéndose además como un proceso que no es de tipo Take, Drop, Attend or Go, asumiéndose que otros procesos resultan más interesantes en un cuento que estos. ValuableAgent especializa el concepto ValuableAgent de *Simulation*, definiéndose además como un agente de uno de estos tipos inferido: Boy, Girl, Kid, Man, Woman, OldPerson o del tipo normalizado Spirit. ValuableObject especializa el concepto ValuableObject de *Simulation*, definiéndose además como un objeto que es arma o armadura. Finalmente ValuableLocation especializa el concepto ValuableLocation de *Simulation*, definiéndose además como un lugar que o bien es de tipo Dwelling o es el origen, mediante la relación isOriginOf de algún objeto valioso.

Con respecto a las operaciones sobre individuos normalizados, puede verse un resumen de la API de este componente en el Cuadro 4.1. El acceso, la modificación, la creación y la destrucción se realizan exclusivamente sobre individuos normalizados.

Acceso: A individuos subtipos de Agent, Object, Location, Process y todos sus individuos conectados, que serán devueltos según el tipo de simulación que les corresponda
Modificación: De individuos subtipos de Agent, Object y especialmente Process, siendo reordenados los procesos respetando el orden temporal
Creación: De individuos subtipos de Agent, Object y Process, pudiendo crear cadenas temporales de estos últimos, y habiendo métodos para crear e incluir automáticamente estados para los agentes y objetos que participan en la creación de un proceso
Destrucción: De individuos subtipos de Agent, Object y Process, ocurriendo un reorganización de las cadenas de procesos que se vean interrumpidas

Cuadro 4.1: Resumen de la API del componente *Tale*

4.1.2. *Propp*

El componente *Propp* es un componente ontológico específico para el desarrollo de esta aplicación.

Este componente contiene la semántica declarativa y operacional del modelo de conocimiento propuesto para representar la narración según la morfología de Vladimir Propp expuesta en el apartado 3.4.5, con sus personajes y funciones de personajes características. Los conceptos de este componente son *concretos*. Este componente es considerado de *nivel bajo*, ya que no se espera que sea especializado por otro componente. Además sus conceptos tienen una jerarquía de *fragmentación media*, heredada del dominio de las historias abstractas, que se encuentra siempre dividido entre narración y simulación. Este componente especializa el componente *AbstractStory*, combinado con *Tale*, centrándose principalmente en especializar conceptos normales de la fábula, asumiendo que el discurso será una secuencia de mensajes trivialmente ordenado de acuerdo a la misma línea temporal que la simulación.

Una vista general de la jerarquía conceptual de este componente se muestra en las Figuras 4.5, 4.6 y 4.7.

Los conceptos normales de este componente son AbstractStory y CharacterFunction. AbstractStory especializa el concepto AbstractStory del componente del mismo nombre, representando una historia abstracta según la morfología proppiana. CharacterFunction especializa el concepto AbstractStory de *Fabula*, representando a las funciones de personaje vistas según Vladimir Propp. Dado que fue este formalista quien propuso por primera vez el concepto de "funciones de personaje", esta jerarquía conceptual cuenta con decenas de conceptos. Los más numerosos son los subtipos de CharacterFunction, llamados MoveFunction (dividido en Preparation, Conflict, DonorMoveFunction y HelperMoveFunction) y ResolutionFunction (dividido en ClaimsOfFalseHero, DifficultTaskWithSolution, ExposureOfFalseHero, Punishment, RecognitionOfHero, Transfiguration, UnrecognizedArrival, WeddingOrThrone). En la vista de la jerarquía conceptual no se despliegan todos los conceptos por razones de espacio y claridad, pero en estas funciones de personaje hay cabida para sucesos narrativos tan dispares como un secuestro de un familiar hasta una boda con ascensión al trono incluida.

La ontología de este componente, que originalmente recibía el nombre de *ProppianOnto*, fue uno de los primeros resultados públicos de este trabajo [PGDA04, PG05b]. Como tal, ha sido adaptada y utilizada en el proyecto *IST* mencionado en el apartado 2.2.5, junta con las primeras versiones de algunas de las ontologías de los otros componentes básicos del armazón propuesto.

Los conceptos no normales descienden de TypedCharacter, TypedProp y TypedSetting. Los descendientes de TypedCharacter, arquetipos de personajes

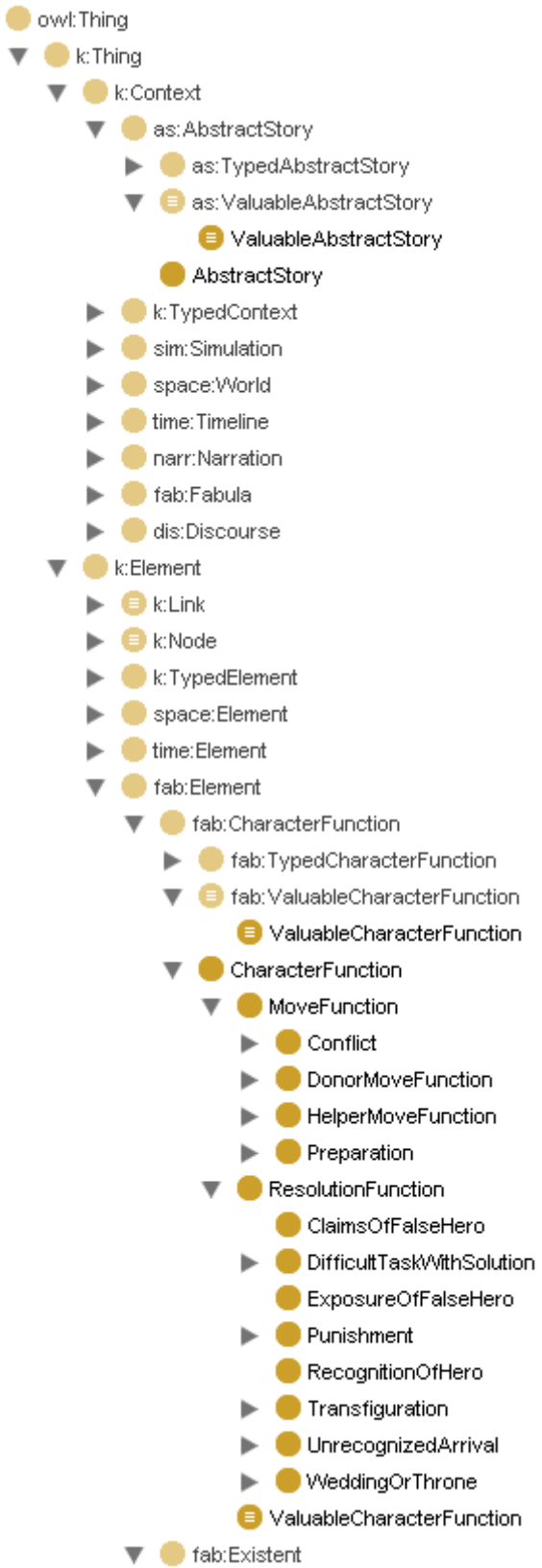

Figura 4.5: Jerarquía conceptual del componente *Propp* (1/3)

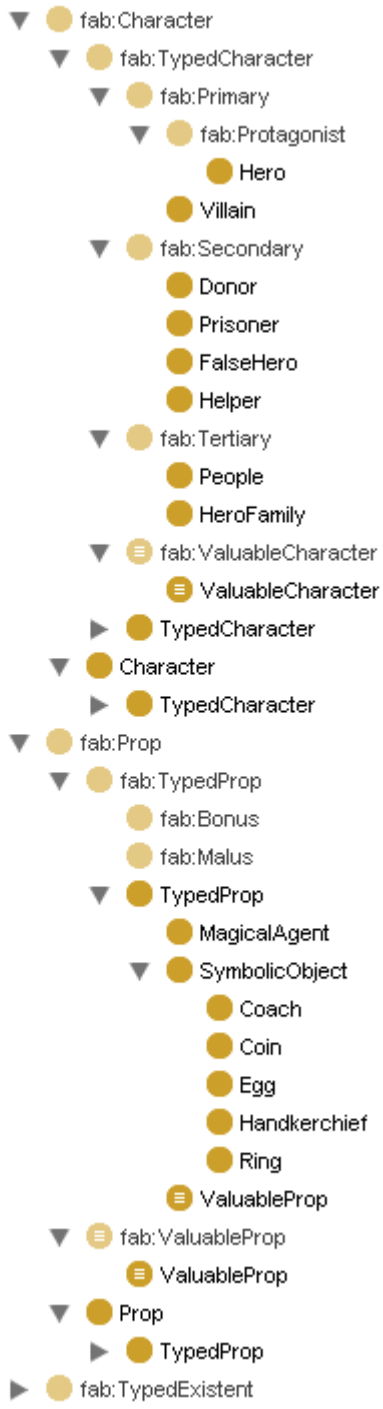

Figura 4.6: Jerarquía conceptual del componente *Propp* (2/3)

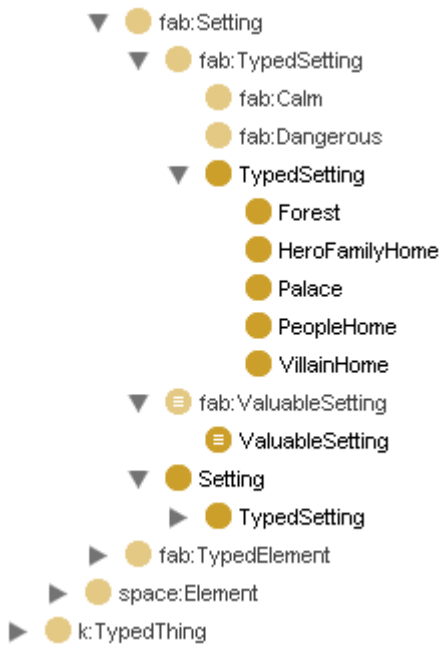

Figura 4.7: Jerarquía conceptual del componente *Propp* (3/3)

clásicos propuestos por Propp para actuar en sus funciones, son Hero, Villain, Donor, Helper, Prisoner, FalseHero, HeroFamily y People. Los descendientes de TypedProp son MagicalAgent y SymbolicObject, que a su vez se divide en tipos como Coach, Coin, Egg, Handkerchief o Ring. Los descendientes de TypedSetting son Forest, HeroFamilyHome, Palace, PeopleHome y VillainHome.

Este componente no añade nuevos roles, ya que se centra sólo en enriquecer el vocabulario conceptual del modelo.

Volviendo a los conceptos, los restantes son definidos y son ValuableAbstractStory, ValuableCharacterFunction, ValuableCharacter, ValuableProp y ValuableSetting. ValuableAbstractStory especializa el concepto ValuableAbstractStory del componente *AbstracStory*, definiéndose además como una historia abstracta valiosa que tiene una fábula con una función de personaje de tipo ValuableCharacterFunction de Propp.ValuableCharacterFunction especializa el concepto ValuableCharacterFunction del componente *Fabula*, definiéndose además con una serie muy numerosa de restricciones que fuerzan a emparejar funciones de personaje que deben ir seguidas, por ejemplo: ser de tipo Preparation y causa directa de otra función de tipo Conflict, o ser de tipo Resolution y no tener causa directa otra función de personaje de tipo MoveFuntion. ValuableCharacter especializa el concepto ValuableCharacter del componente *Fabula*, definiéndose además como un personaje que es protagonista a la vez que héroe proppiano, personaje primario a la vez que villano, o bien personaje

secundario y a la vez alguno de los otros tipos distintos al héroe y al villano. ValuableProp especializa el concepto ValuableProp del componente *Fabula*, definiéndose además como accesorio de tipo SymbolicObject o MagicalAgent. ValuableSetting especializa el concepto ValuableSetting del componente *Fabula*, definiéndose además como lugar que no es bosque, asumiendo que son lugares de paso en la mayor parte de los cuentos analizados por Propp.

Con respecto a las operaciones sobre individuos normalizados, puede verse un resumen de la API de este componente en el Cuadro 4.2. El acceso, la modificación, la creación y la destrucción se realizan exclusivamente sobre individuos normalizados.

Acceso: A individuos AbstractStory, Character, Setting, Prop, CharacterFunction y todos sus individuos conectados, que serán devueltos según el tipo que les corresponda
Modificación: De individuos AbstractStory, Character, Setting, Prop y CharacterFunction, siendo reordenadas las funciones de personaje respetando el orden causal
Creación: De individuos AbstractStory, Character, Setting, Prop y CharacterFunction, pudiendo crear cadenas causales de estos últimos
Destrucción: De individuos AbstractStory, Character, Setting, Prop y CharacterFunction, ocurriendo un reorganización de las cadenas de funciones de personaje

Cuadro 4.2: Resumen de la API del componente *Propp*

4.1.3. *KICBR*

El componente *KICBR* es un componente ontológico específico para el desarrollo de esta aplicación.

Este componente contiene la semántica declarativa y operacional de un sistema de razonamiento basado en casos intensivo en conocimiento (KI-CBR), con los casos, los problemas, las soluciones y los distintos elementos que estos sistemas utilizan. Este componente no modela un dominio de conocimiento, sino un metadominio con individuos que no representan ninguna realidad, sino que se utilizan internamente como apoyo a las tareas de razonamiento. Los conceptos de este componente son *concretos*. Este componente es considerado de *nivel bajo*, ya que no está contemplado que otro componente lo especialice. Además sus conceptos tienen una jerarquía de *fragmentación baja*, dado que de ser especializado no es fácilmente divisible en partes distintas. Este componente especializa al componente *Knowledge*.

Una vista general de la jerarquía conceptual de este componente se muestra en la Figura 4.8.

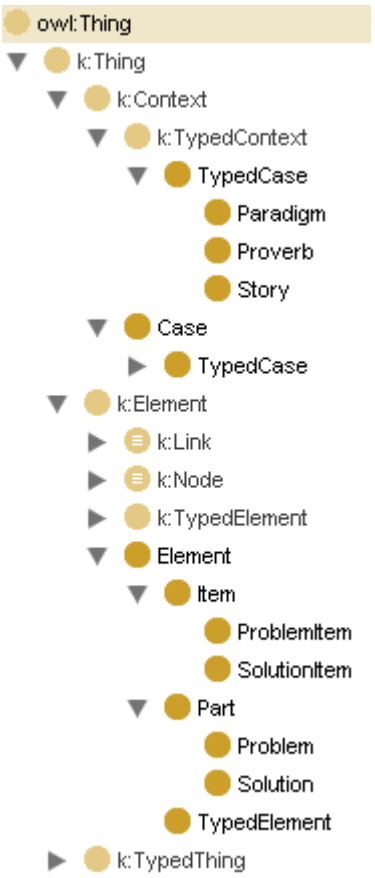

Figura 4.8: Jerarquía conceptual del componente *KICBR*

Los conceptos normales de este componente son Case, Problem, Solution, ProblemItem y SolutionItem. Case representa un caso dentro del sistema KI-CBR. Problem representa el problema descrito en un caso del sistema KI-CBR. Solution representa la solución a un problema descrito en un caso del sistema KI-CBR. Finalmente ProblemItem y SolutionItem son los elementos que se utilizan para describir el problema y la solución, respectivamente, de un caso del sistema KI-CBR. Estos conceptos están inspirados en las ideas de la ontología *CBROnto* del proyecto *COLIBRI*, así como la implementación del sistema KI-CBR mediante DLs aprovecha ideas ya presentadas en el apartado 4.1.3.

Los conceptos no normales son descendientes de TypedCase y son Paradigm, Proverb e Story, distinguiendo entre los casos paradigmáticos, prover-

biales y de tipo "historia" según las ideas de Roger Schank presentadas en el apartado .

La jerarquía de roles de este componente se muestra en la Figura 4.9.

Los roles de este componente son isCaseOfProblem, isCaseOfSolution, isSuperCaseOf, hasProblemItem, hasSolutionItem, directlyDependsOn y sus respectivas inversas. Los roles isCaseOfProblem e isCaseOfSolution sirven para conectar un caso con las dos partes de que se compone: el problema y la solución. El rol isSuperCaseOf y su inversa sirven para crear jerarquías de casos en forma de árbol, siendo los padres casos más generales que actúan como metacontextos de sus hijos, casos más específicos. El rol hasProblemItem sirve para relacionar un problema con sus elementos, y el rol hasSolutionItem hace lo mismo pero con las soluciones. Finalmente el rol directlyDependsOn sirve para conectar un individuo de tipo SolutionItem con el individuo de tipo ProblemItem o SolutionItem del que depende. Si un elemento de la solución depende de otro del problema, quiere decir que en caso de que surja un nuevo problema con dicho elemento, la nueva solución requiere el elemento de la solución señalado. Si un elemento de la solución depende de otro, quiere decir que en caso de utilizarse dicha solución para la adaptación, si se añade o elimina el segundo elemento hay que añadir o eliminar el primero.

Este componente no utiliza conceptos definidos.

Con respecto a las operaciones sobre individuos normalizados, puede verse un resumen de la API de este componente en el Cuadro 4.3. El acceso, la modificación, la creación y la destrucción se realizan exclusivamente sobre individuos normalizados.

Acceso: A individuos Case, Problem, Solution, ProblemItem y SolutionItem y todos sus individuos conectados, que serán devueltos según el tipo que les corresponda **Modificación:** De individuos Case, Problem y Solution **Creación:** De individuos Case, Problem, Solution, ProblemItem y SolutionItem **Destrucción:** De individuos Case, Problem, Solution, ProblemItem y SolutionItem, sin destruir contenido específico de dominio al destruir el problema o la solución

Cuadro 4.3: Resumen de la API del componente *KICBR*

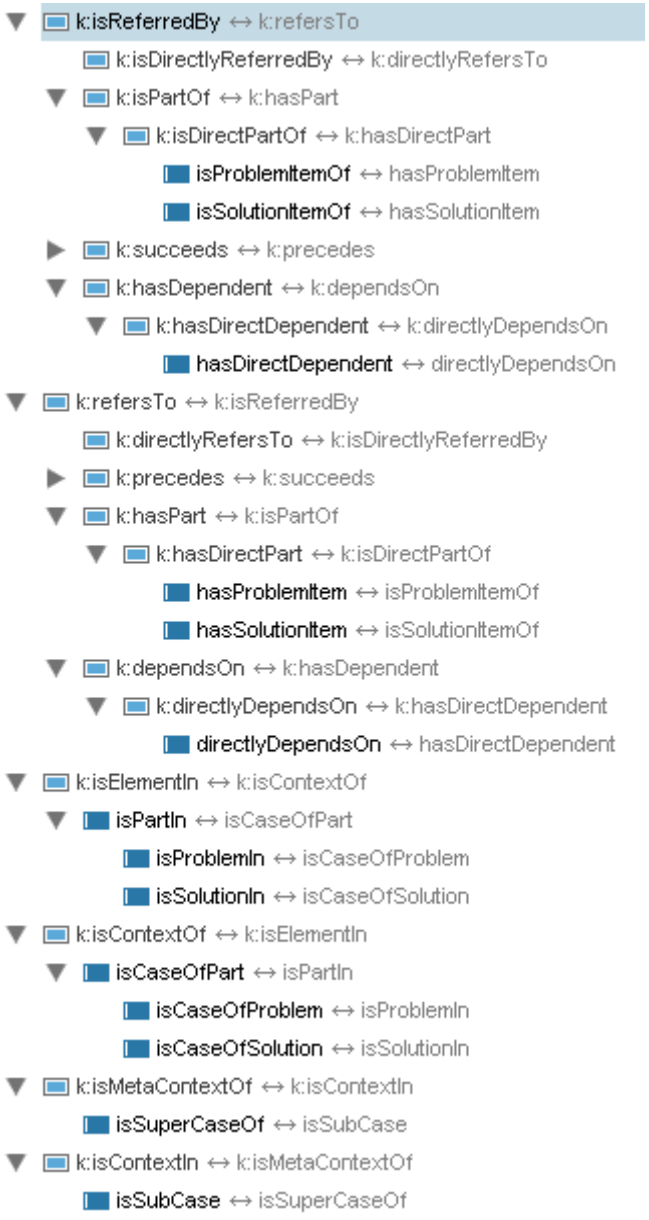

Figura 4.9: Jerarquía de roles del componente *KICBR*

4.2. Funcionamiento de la aplicación

Lo primero que se aparece al ejecutar la aplicación es una interfaz única dividida en tres partes: un formulario para que el usuario introduzca la solicitud inicial con lo que desea pedirle al sistema, un cuadro de texto donde se muestran las historias obtenidas y los botones con las posibles acciones a realizar.

En la Figura 4.10 se muestra la interfaz de la aplicación *ProtoPropp* mientras el usuario establece la solicitud inicial para la aplicación. Esta solicitud contiene los elementos esenciales de un cuento, que servirá como semilla para el resto de las operaciones que se realicen con la aplicación. Este cuento semilla no está completo, de hecho, incluso puede estar completamente vacío si el usuario así lo desea. El formulario de la interfaz presenta varias cajas de selección con los tipos de personajes posibles que existen en el modelo cargado, en este caso los del componente *Propp*. Además hay más cajas de selección para los tipos principales de funciones de personaje.

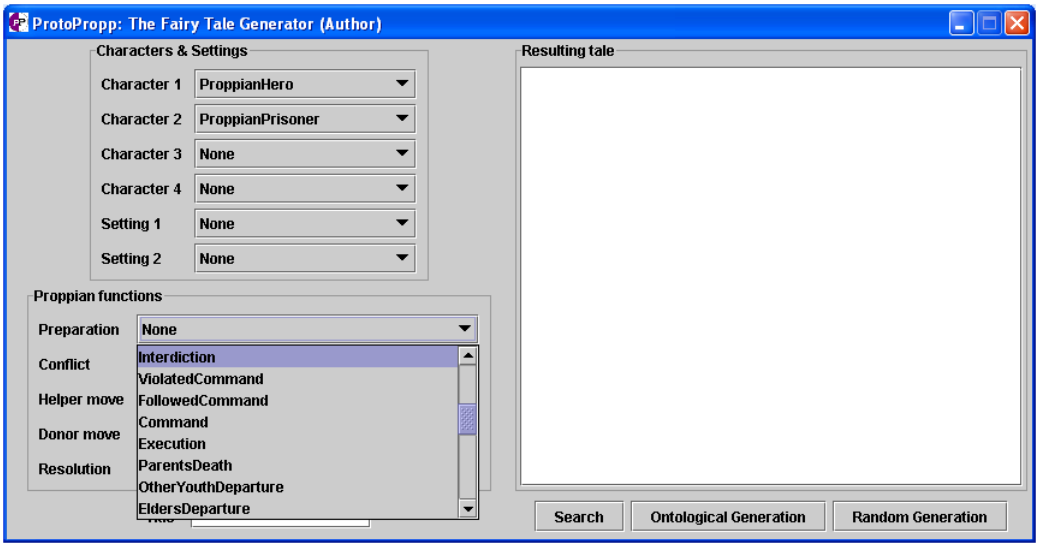

Figura 4.10: Estableciendo la solicitud inicial para *ProtoPropp*

ProtoPropp instancia el algoritmo general propuesto en el apartado 3.5. La aplicación final tiene tres modos de uso: búsqueda en la base de casos, generación de un cuento por el procedimiento ontológico y generación de un cuento por un procedimiento aleatorio.

La *búsqueda* funciona de la siguiente manera: el usuario introduce una consulta que pretende representar las restricciones de un cuento maravilloso, después se produce una fase de recuperación donde se extraen los cuentos

de la base de casos más similares a la consulta del usuario, y por último se convierten dichos cuentos a lenguaje natural mediante plantillas para así poder mostrarlos por pantalla.

En la Figura 4.11 se muestra una historia recuperada directamente de la base de conocimiento, usando la pantalla principal de la interfaz.

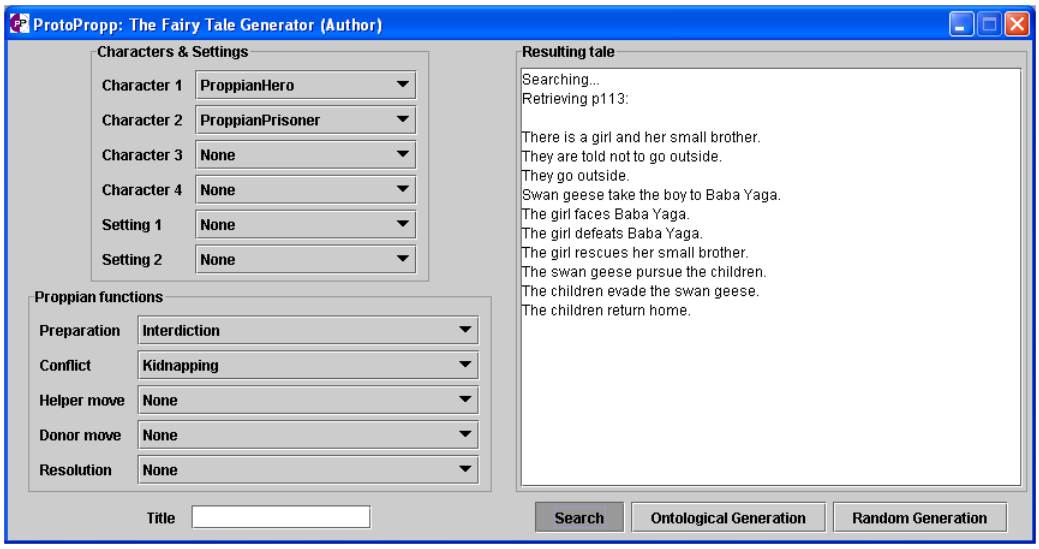

Figura 4.11: Buscando una fábula en el corpus de *ProtoPropp*

La *generación aleatoria* de un cuento tiene utilidad como referencia para comparar la eficacia del generador ontológico, que es en realidad la operación más interesante de *ProtoPropp*. El generador aleatorio escoge un número de movimientos (entre el mínimo y el máximo que tengan los casos de la base de casos), un número de funciones de personaje proppianas (también entre el mínimo y el máximo de los casos de la base de casos) y asigna personajes a esas funciones aleatoriamente escogiendo entre si ejercen un rol activo o pasivo, habiendo desactivado previamente la mayoría de las restricciones de la ontología.

En la Figura 4.12 se muestra una historia aleatoria generada en *ProtoPropp*, tal y como lo ve el usuario a través de la interfaz de la aplicación.

La *generación ontológica* de un cuento nuevo tiene muchas partes en común con la búsqueda de cuentos en la base de casos, siendo un proceso de razonamiento basado en casos que ya ha sido suficienmente documentado [DAGP04, GDAPH04, GDAPH05]. Una vez recuperados los cuentos más similares a la solicitud del usuario, pueden ocurrir tres cosas:

1. Existe al menos un cuento recuperado que respeta todas las restricciones de la solicitud. En este caso se informa al usuario de que no

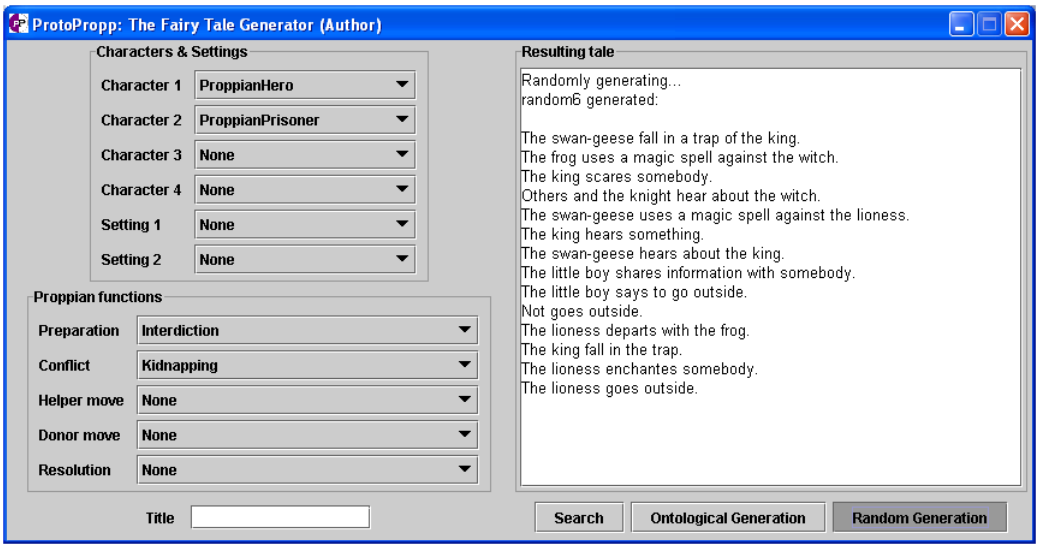

Figura 4.12: Generando una fábula aleatoria en *ProtoPropp*

es posible crear un cuento nuevo si no se añaden más restricciones, sugiriendo la posibilidad de añadirlas aleatoriamente.

2. Existe al menos un cuento recuperado que respeta alguna restricción de la solicitud. En este caso se escoge el cuento recuperado que más restricciones cumple de manera exacta, se elabora una lista de las restricciones que no se cumplen de manera exacta y se buscan cuentos que sí cumplan esas restricciones. Esta segunda fase de recuperación permite que se relajen las restricciones todo lo posible, de manera que siempre se recuperará algún cuento. Una vez recuperado otro cuento, se realiza una adaptación, sustituyendo las nuevas "restricciones" encontradas a las del primer cuento, para crear uno nuevo, inventándose todas las necesarias hasta completarlas todas. Luego se convierte a lenguaje natural el resultado, usando plantillas, y se muestra por pantalla.

3. No existe un cuento recuperado que respete alguna de las restricciones. En este caso se informa al usuario de que no es posible crear un cuento nuevo si no se cambian o añaden más restricciones.

En la Figura 4.13 se muestra una historia generada mediante *ProtoPropp*.

La recuperación se basa en la similitud estructural de los conceptos. En principio se intenta recuperar conceptos idénticos, pero admite hasta 1 grado de ascendencia (conceptos padres) y n grados de descendencia. El cuento recuperado sirve como boceto para hacer la adaptación. Se localizan las *carencias*, componentes del cuento que aparecen en la solicitud del usuario

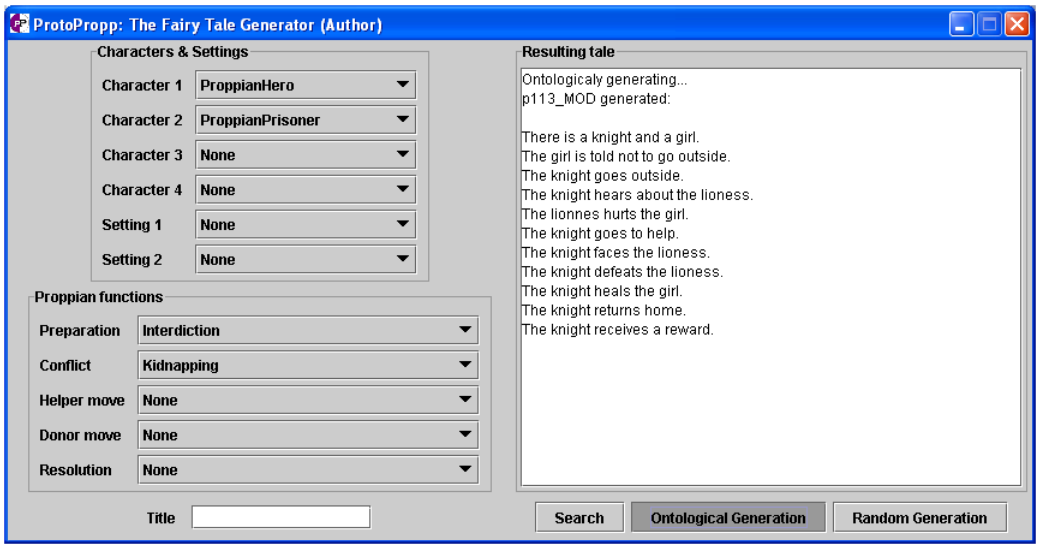

Figura 4.13: Generando una nueva fábula mediante *ProtoPropp*

pero no en el cuento recuperado, y los *sobrantes*, componentes del cuento que aparecen en el cuento recuperado pero no en la solicitud del usuario.

La adaptación trivial consiste en seleccionar un movimiento del cuento recuperado y eliminar un número aleatorio m de funciones de personaje proppianas (entre 1 y f, siendo f el número total de funciones del movimiento). Después se eliminan todas las funciones que dependieran de las eliminadas, en una cadena recursiva, y se añaden otras m funciones aleatoriamente, añadiendo también todas las funciones de las que estas funciones dependan. Los personajes de las funciones añadidas serán tomados del reparto de personajes existentes, y en caso de no ser posibles se crean nuevos, pero siempre manteniendo las restricciones de participación de las funciones.

Tanto las carencias como los sobrantes pueden disparar más ciclos de recuperación para obtener otros cuentos con los que mezclar el primero y realizar así la adaptación. Las carencias en funciones proppianas o en personajes se cubren con funciones proppianas o personajes de otros cuentos, teniendo en cuenta que no existan incompatibilidades y pudiendo sustituir sobrantes si fuese necesario Para el caso de las funciones, se añadirán también todas aquellas que tengan una relación de dependencia con las que han sido incorporadas. Los sobrantes que resten tras ese proceso se cambian aleatoriamente por instancias de los mismos conceptos, para dar variabilidad al cuento.

4.3. Evaluación de la aplicación

Como parte de esta validación del armazón propuesto, además de las configuraciones y pruebas por defecto de los componentes básicos y específicos disponibles a través del apéndice A, se ha realizado un experimento para evaluar *ProtoPropp* de forma empírica en función de la impresión que causan las historias abstractas que genera. En este experimento [PG06a] se generan tres historias partiendo de una solicitud inicial vacía, aunque habiendo configurado la aplicación de tres maneras muy diferentes, con la intención de comprobar qué tal es el comportamiento del algoritmo de generación propuesto, y por lo tanto probar que el armazón sirve para desarrollar aplicaciones de narración automática.

Las historias de los Cuadros 4.4, 4.5 y 4.6 fueron evaluadas en un proceso ciego, dado que los usuarios sabían que se trataban de historias generadas por ordenador pero desconocían su procedencia exacta. La primera historia procede directamente del corpus de cuentos maravillosos incluido en *ProtoPropp*, por lo que ha sido editada completamente a mano. Concretamente es la fábula simplificada de *The Swan Geese*, el cuento número 113 de la colección de Afanasiev. La segunda historia fue generada aleatoriamente, eliminando la mayoría de las restricciones de las ontologías de los componentes involucrados en la generación. La tercera historia fue generada mediante *ProtoPropp*, usando las ontologías completas de todos los componentes involucrados, incluyendo el metadominio *KICBR* y el razonamiento basado en DLs que se ha explicado en el capítulo 3.

Once upon a time... there was the girl and the little boy. The father said to the girl not to go outside. The girl went outside. The swan-geese kidnapped the little boy. The girl departured. The girl and the witch started a duel. The girl won the duel. The girl rescued the little boy. The swan-geese tested the girl. The girl passed the test. The girl returned home with the little boy.

Cuadro 4.4: Historia del corpus incluido en *ProtoPropp*

En el Cuadro 4.7 se muestran los resultados de aplicar las métricas formales de los apartados 3.6.2 y 3.6.3 para calcular el valor y la novedad de las tres historias.

Tras esta primera evaluación formal, el método empírico para evaluar estas historias consistió en enviar por correo electrónico un cuestionario a 48

Once upon a time... the swan-geese fell in the trap of the king. The frog used a magic spell against the witch. The king scared somebody. Others and the knight heard about the witch. The swan-geese used a magic spell against the lioness. The king heard something. The swan-geese heard about the king. The little boy shared information with somebody. The little boy said to go outside. Not went outside. The lioness departed with the frog. The king fell in the trap. The lioness enchanted somebody. The lioness went outside.

Cuadro 4.5: Historia generada aleatoriamente en *ProtoPropp*

Once upon a time... there was a princess. The princess said not to go outside. The princess went outside. The princess heard about the lioness. The lioness scared the princess. The lioness kidnapped the princess. The knight departured. The knight and the lioness fought. The knight won the fight. The knight solved the problem of the princess. The knight returned. A big treasure to the knight.

Cuadro 4.6: Historia generada con todos los componentes de *ProtoPropp*

Métrica	*Corpus*	*Aleatoria*	*Normal*
Valor	0.77	0.12	0.59
Novedad	0.0	0.43	0.33

Cuadro 4.7: Resultados de aplicar las métricas formales

usuarios formados por estudiantes y compañeros de la Facultad con al menos un nivel básico de conocimiento de la lengua inglesa. Las instrucciones que recibieron estos jueces aparecen en el Cuadro 4.8:

1. This questionnaire is an anonymous evaluation of **stories generated by computer programs**. The short texts presented below are **not stories** but **basic stories**, one of the steps in the development process of a more complex and complete story. Probably you will have to infer some missing information.

2. Please, read the basic stories carefully and put a mark on your answers (no more or less than **one answer for each question**). This is the explanation for the evaluation factor that you will be asked about:

a. **Linguistic Quality** means... "How is the text?"
(e.g. 0 = Illegible, 5 = Acceptable, 10 = Wonderful)

b. **Coherence** means... "How is the sequence of events?"
(e.g. 0 = Absurd, 5 = Reasonable, 10 = Perfectly linked)

c. **Interest** means... "How is the story for you?"
(e.g. 0 = Boring, 5 = Satisfactory, 10 = Exciting)

d. **Originality** means... "How is the plot compared to others?"
(e.g. 0 = Identical, 5 = Different, 10 = Unique)

3. When evaluating kindly take into account that computer program's final goal is to generate simple traditional folk-tales.

Cuadro 4.8: Instrucciones para los jueces del experimento

En este estudio la escala numérica utilizada fue la siguiente: los usuarios puntuaban cada factor de satisfacción con la historia con un número entre 0 y 10. Los factores evaluados fueron: *calidad lingüística*, *coherencia*, *interés* y *originalidad*. Los resultados de las encuestas se muestran en los Cuadros 4.9, 4.11, 4.13 y 4.15, con las frecuencias de las distintas respuestas, y en los Cuadros 4.10, 4.12, 4.14 y 4.16, con los estadísticos más importantes. En el apartado 5.4 se discuten ampliamente estos resultados.

Calidad lingüística	*Corpus*	*Aleatoria*	*Normal*
Puntuación 0	0	0	0
Puntuación 1	3	10	3
Puntuación 2	3	6	1
Puntuación 3	10	5	7
Puntuación 4	5	8	5
Puntuación 5	8	6	8
Puntuación 6	7	4	7
Puntuación 7	7	3	12
Puntuación 8	4	6	5
Puntuación 9	1	0	0
Puntuación 10	0	0	0

Cuadro 4.9: Frecuencias sobre la calidad lingüística

Calidad lingüística	*Corpus*	*Aleatoria*	*Normal*
Media	4.81	4.00	5.25
Desviación típica	2.09	2.37	1.99

Cuadro 4.10: Estadísticos sobre la calidad lingüística

Coherencia	*Corpus*	*Aleatoria*	*Normal*
Puntuación 0	1	13	0
Puntuación 1	1	11	2
Puntuación 2	5	12	0
Puntuación 3	8	5	0
Puntuación 4	8	3	1
Puntuación 5	6	1	5
Puntuación 6	5	2	7
Puntuación 7	7	0	8
Puntuación 8	5	1	10
Puntuación 9	2	0	10
Puntuación 10	0	0	3

Cuadro 4.11: Frecuencias sobre la coherencia

Coherencia	*Corpus*	*Aleatoria*	*Normal*
Media	4.88	1.81	6.94
Desviación típica	2.24	1.82	2.24

Cuadro 4.12: Estadísticos sobre la coherencia

Interés	*Corpus*	*Aleatoria*	*Normal*
Puntuación 0	2	6	0
Puntuación 1	3	8	2
Puntuación 2	6	5	1
Puntuación 3	6	10	9
Puntuación 4	8	6	5
Puntuación 5	10	2	8
Puntuación 6	4	5	12
Puntuación 7	6	5	7
Puntuación 8	2	1	4
Puntuación 9	1	0	0
Puntuación 10	0	0	0

Cuadro 4.13: Frecuencias sobre el interés

Interés	*Corpus*	*Aleatoria*	*Normal*
Media	4.29	3.23	5.08
Desviación típica	2.17	2.32	1.83

Cuadro 4.14: Estadísticos sobre el interés

Originalidad	*Corpus*	*Aleatoria*	*Normal*
Puntuación 0	1	2	0
Puntuación 1	4	5	6
Puntuación 2	3	0	7
Puntuación 3	4	3	6
Puntuación 4	7	3	11
Puntuación 5	4	8	7
Puntuación 6	9	6	7
Puntuación 7	11	12	3
Puntuación 8	4	7	1
Puntuación 9	1	1	0
Puntuación 10	0	1	0

Cuadro 4.15: Frecuencias sobre el interés

Originalidad	*Corpus*	*Aleatoria*	*Normal*
Media	5.04	5.44	3.92
Desviación típica	2.28	2.50	1.88

Cuadro 4.16: Estadísticos sobre la originalidad

Capítulo 5

Discusión

Defenderé mis opiniones hasta la muerte,
pero daré mi vida para que vos podáis defender las vuestras.
Voltaire

En este capítulo se ofrece una discusión detallada sobre este trabajo de investigación. El propósito de esta discusión es averiguar si el material y método de investigación han sido los adecuados para perseguir los objetivos propuestos en el apartado 1.4, así como explicar el significado de los resultados presentados en el capítulo 4 en función de lo anterior.

A lo largo de la discusión aparecen algunas referencias al apartado 2.2 con la intención de comparar, siempre que sea posible, el material y método de investigación propuestos con los de otros proyectos de narración automática, así como los resultados de este trabajo con los obtenidos en otros trabajos sobre aplicaciones de narración automática.

Se comienza discutiendo los aspectos fundamentales del material y método de investigación escogidos para realizar esta investigación, poniendo de manifiesto sus ventajas e inconvenientes en comparación con las características de los otros proyectos. Finalmente se comparan y discuten también los resultados obtenidos en este proyecto frente a los de proyectos anteriores.

5.1. Representación de historias

Como se ha puesto de manifiesto en este trabajo uno de los aspectos más importantes de las aplicaciones de narración automática es la representación de conocimiento, un importante cuello de botella para su desarrollo.

El armazón propuesto apuesta claramente por el formalismo de las lógicas descriptivas como semántica declarativa estándar para el desarrollo de todos los componentes de dominio de una aplicación. La expresividad de

este formalismo permite ofrecer un conjunto bastante amplio de componentes básicos, que representan los dominios de conocimiento que necesita una aplicación de narración automática típica. El hecho de que las DLs sean la evolución de los sistemas de marcos y redes semánticas, tan utilizadas en los proyectos pioneros de narración automática, explica que ahora sean capaces de expresar la mayor parte de los conocimientos que se representaban antes.

Comparando el formalismo de las DLs con otros sistemas de representación más populares como los marcos de Minstrel [PG06b], implementados en *Lisp*, se encuentran diferencias notables que decantan la balanza de la reutilización a favor del primero [PG06b]. No se trata de una cuestión de flexibilidad, ya que la potencia expresiva de *Lisp* es superior al tratarse de un lenguaje de propósito general muy extensible, sino de ofrecer un formalismo que permita hacer algo que todavía no se ha conseguido en el campo de la narración automática: desarrollar una aplicación por componentes y entre varios desarrolladores; para lo cual es requisito imprescindible sacrificad libertad de representación a cambio de tener una semántica explícita para cada una de las partes en que se divida el sistema.

Evidentemente la decisión de utilizar DLs ha condicionado de manera importante el enfoque de este trabajo, y como se explica en el siguiente apartado, no todo han sido ventajas. Aún así, la elección de un punto de vista concreto y formal sobre la representación de conocimiento ha sido de gran utilidad para comprender mejor las diferencias entre los proyectos del estado del arte, al mismo tiempo que para identificar las cuestiones fundamentales de la representación de historias.

El lenguaje empleado para implementar este tipo de representación de conocimiento es OWL DL. Este lenguaje está siendo ampliamente utilizado hoy día, debido a su estandarización por parte del W3C como lenguaje de descripción de ontologías para la web. Aunque este lenguaje está fundamentado sobre un formalismo estable, una familia concreta de las lógicas descritivas, lo cierto es que está siendo remodelado y ya existe una versión actualizada del mismo. Esto, junto a la inmadurez de las herramientas de edición y razonamiento que se menciona en el siguiente apartado, son pruebas de que OWL DL tiene dos inconvenientes para ser utilizado hoy día en aplicaciones comerciales. En primer lugar aún tienen que documentarse más casos de reutilización de ontologías OWL DL en proyectos de gran envergadura para consolidarse como lenguaje estándar de descripción de ontologías. La premisa del mundo abierto, el no asumir nombres únicos en los individuos de la base de conocimiento, etc. son obstáculos para el diseñador tradicional de ontologías y bases de conocimiento, y aún no se ha demostrado que -incluso usando técnicas de normalización- sea posible escalar el diseño de sistemas basados en OWL DL. En segundo lugar, para tareas complejas de razona-

miento como son las requeridas por una aplicación de narración automática, OWL DL necesita verse complementado con otros lenguajes que aumenten su expresividad. SWRL, y concretamente las reglas seguras para lógicas descriptivas, son un paso acertado en aumento de la expresividad y conservación de las buenas propiedades computacionales de OWL DL. Sin embargo, dado el rendimiento de las implementaciones actuales de razonamiento con SWRL, ha sido necesario prescindir de estas reglas en los componentes básicos del armazón propuesto.

En este trabajo no se han considerado ciertas restricciones de bajo nivel relativas a las técnicas de presentación de las historias para poder prestar más atención a cuestiones más relevantes de los proyectos analizados.

La adquisición de conocimiento es siempre un cuello de botella en los sistemas basados en conocimiento. Según el modelo propuesto, los componentes ontológicos y las bases de conocimiento forman parte de la obra de los desarrolladores de aplicaciones de narracion automática, por lo que el sistema los toma como parte reusable y con entidad propia del producto y no como parte del problema.

El proyecto *IST* [GBPMRI07] puede se considerado un resultado preliminar del armazón propuesto, dado que usa la ontología del componente *Propp* para generar cuentos maravillosos en forma de textos sencillos basados en plantillas, de forma similar a como lo hace *ProtoPropp*, solo que este último sí utiliza el armazón al completo. Sin embargo es importante señalar que *IST* no utiliza ningún concepto declarado ni definido, por lo que prácticamente se desaprovecha todo el potencial de las DLs, aunque puede considerarse un primer paso hacia aplicaciones más avanzadas que sí lo aprovechen. Por otra parte el trabajo resulta interesante debido al enfoque CBR que propone, más completo que el *ProtoPropp* ya que cuenta con dos tipos diferentes de adaptación de soluciones, una convencional y otra constructiva en la que interviene además un algoritmo de búsqueda en grafos que se utiliza para recorrer la estructura de los individuos relacionados de la base de conocimiento.

Una cuestión discutible es el tratamiento de los diferentes dominios que intervienen en la generación de una historia. Estos dominios pueden ser tratados de forma independiente o uniforme. En el caso de la fábula, el discurso y la presentación son muchos los sistemas que los tratan de forma separada. Sin embargo también hay opiniones contrarias a esa separación. Por ejemplo Birte Lönneker [L04] señala que el principal error en la arquitectura de *Brutus* es precisamente ese intento de construir narrativa desarrollando la fábula y el discurso de forma completamente independiente, uniendo ambos productos sólo al final del proceso.

Para construir una historia se necesita además, conocimiento que no necesariamente se encuentra representado de forma explícita en el producto

final, sino que se utiliza de forma auxiliar para guiar el proceso de generación y narración de la historia, así como para definir las heurísticas necesarias para generar y evaluar los resultados. Aunque evidentemente existen otros recursos de conocimiento, como reglas o patrones parecidos a los 24 TRAMs (Transform-Recall-Adapt-Methods) que incluye Turner en su *Minstrel*, pensados para dotarlo de creatividad.

5.2. Generación de historias

Aunque se ha hablado mucho sobre la capacidad de representación de las DLs, es importante señalar que las DLs tienen serias limitaciones a la hora de establecer mecanismos de inferencia complejos. Esto ocurre así en muchas ocasiones, como advierten Davis, Shrobe y Szolovits.

> Expressing reasoning strategies in first order logic is in keeping with the spirit of the logicist approach, viz., explicit representation of knowledge in a uniform, declarative representation. But this is often problematic in practice: a language designed to express facts declaratively is not necessarily good for expressing the imperative information characteristic of a reasoning strategy. [DSS93]

Una desventaja común a numerosos sistemas es que utilizan gran cantidad de conocimiento estructurado con un formato propio. Este es el caso de *Minstrel* que utiliza redes semánticas. Además pocas veces nos han llegado las aplicaciones desarrolladas en estos proyectos, y mucho menos su código fuente. Este conocimiento no resulta mantenible a largo plazo y por eso en este trabajo se emplean ontologías implementadas mediante DLs, además de un enfoque de Ingeniería del Software para potenciar la mantenibilidad y reusabilidad del código y de los ejecutables de las aplicaciones desarrolladas.

Aunque la aplicación de ejemplo utiliza KI-CBR, a lo largo del trabajo se ha evitado deliberadamente comprometerse con un enfoque de generación determinado, como podrían ser las gramáticas de producción o las técnicas de planificación. En general el armazón puede verse enriquecido si los componentes añaden funciones de resolución de problemas a sus APIs, de manera que no todo el peso de la generación recaiga en los desarrolladores de la aplicación.

La escalabilidad, no sólo en cuanto a representación de historias grandes y complejas, sino en cuanto al razonamiento sobre ellas, es uno de los problemas más interesantes y retadores para los diseñadores de un armazón. En general los sistemas de razonamiento basado en reglas suelen padecer mayores problemas de escalabilidad que los sistemas de razonamiento basado en

casos, aunque los segundos tienen su cuello de botella en la adquisición de conocimiento. El reto del armazón consiste en ofrecer un algoritmo base, en este caso ligado al razonamiento con lógicas descriptivas que resulte escalable a la vez que abierto a las técnicas de cualquier tipo que deseen implementarse por encima. Precisamente las limitaciones de los enfoques donde se combina el CBR con las DLs [GAGCDAFC99] son las que aplican a ProtoPropp. En general, la recuperación de datos es mucho menos eficiente sobre una ontología que sobre una base de datos, pero evidentemente existe un valor añadido que debe ser correctamente explotado para que la perdida en eficiencia compense el conjunto del razonamiento.

Java ha sido elegido como lenguaje de programación de *DLModel*, *DLApplication* y su aplicación de ejemplo, así como el lenguaje de implementación de la semántica operacional de los componentes ontológicos desarrollados para ilustrar esta propuesta. Esta elección se fundamenta en la existencia de numerosas herramientas útiles, basadas en este lenguaje, para desarrollar aplicaciones con tecnologías semánticas; además de en la existencia de extensas comunidades de usuarios especializadas en muchas y diversas librerías. Consideramos esta una buena decisión, que se corrobora con la aparición de aplicaciones como *Wide Ruled* enteramente basadas en *Java*. Una razón fundamental que justifica el éxito de este lenguaje en materia de narración automática es que en el desarrollo de prototipos y experimentos académicos resulta más útil la capacidad de reutilizar el código y su documentación que la de desarrollar un software muy eficiente u optimizado.

Para los procesos de razonamiento más complicados se utiliza *Pellet* que, aunque sin duda es una herramienta fundamental para trabajar con DLs, no ofrece la misma flexibilidad que otros sistemas como *Lisp* y *Prolog*, lo que hace complicado encontrar soluciones a problemas básicos de razonamiento, aunque por otro lado obliga a ser más estricto en la definición de las operaciones semánticas que necesita realizar cada componentes ontológico.

Entrando ya a debatir sobre los distintos métodos de generación de historias, es obligatorio reconocer que *Minstrel*, como sistema KI-CBR creativo de generación de historias, es uno de los principales referentes metodológicos de este trabajo. Si bien este trabajo propone un armazón de desarrollo de aplicaciones, y no un algoritmo ni un modelo concreto de creatividad computacional, es inevitable que en esta discusión se mencionen la cuestión al ser un tema muy ligado a la generación de historias [PG05a].

David Herman [Her] expone la profunda relación existente entre la narratología y las ciencias cognitivas, argumentando que según las definiciones de la *Enciclopedia MIT de las Ciencias Cognitivas* [MIT01], la primera estaría contenida dentro de la segunda. Según Herman, la *proyección parabólica* mencionada en *The Literary Mind* [Tur98], que consiste en relacionar historias

familiares con las nuevas situaciones que nos encontramos cada día, puede entenderse como una analogía del CBR que nos ayuda a entender mejor las etapas de su ciclo.

Una de las conclusiones a las que llega Turner en su trabajo es que para generar una historia sencilla basta con hacer un uso inteligente y "creativo", usando procesos de memoria, planificación y resolución de problemas, etc., de una pequeña cantidad de información. En el caso de *Minstrel* es necesario bastante conocimiento semántico sobre las moralejas, el universo de los caballeros del rey Arturo y las técnicas de narración y presentación del lenguaje, pero no se utiliza demasiado conocimiento episódico en forma de casos en la base de casos.

Al mismo tiempo reconoce que los procesos implementados son sencillos; el propio proceso CBR trata de asemejarse a la forma en que funciona la mente humana. El concepto de TRAM aborda el tema de la creatividad y responde al reto de generar nuevo conocimiento con bastante soltura. Y por último, la memoria imaginativa propone una elegante integración de la creatividad en el sofisticado modelo cognitivo necesario para generar historias.

El proyecto *IST* también utiliza CBR para la generación de sus historias. La recuperación se basa en una similitud muy directa, métrica que en el apartado 3.6.3 ha sido definida con mayor precisión. Tanto el algoritmo convencional de adaptación transformacional como la adaptación constructiva que proponen las autoras encajan bastante bien en el armazón propuesto, siendo incluso posible, como se muestra en *ProtoPropp* enriquecer el proceso con las detecciones de inconsistencias lógicas en el modelo.

Mexica [PyPS01] se presenta como un modelo diferente de generación de historias, basado en ciclos de actividad y reflexión. Estas dos etapas del proceso de generación creativa de una historia son claves para interpretar adecuadamente el funcionamiento de muchos otros modelos, que tienen procesos de revisión integrados en la generación pero que no llegan a formularse de forma tan explícita como dos etapas alternas dentro un ciclo iterativo. Es precisamente este ciclo, lo que hace que el enfoque no pueda ser fácilmente clasificado bajo los paradigmas estructuralista y transformacional, ni tampoco agrupado con las gramáticas de producción o como un método más de resolución de problemas, y que por lo tanto merezca una consideración especial como contribución al estado del arte.

Las llamadas recursivas de la recuperación CBR de *Minstrel* son una primera implementación de una generación iterativa planteada desde el punto de vista del autor. Como Turner [Tur92] sugiere, puede que la creatividad se base en heurísticas simples; pero a esta afirmación hay que añadirle dos grandes inconvenientes. El primero es que, como el mismo Turner reconoce, esas heurísticas pueden ser distintas según el dominio de conocimiento

del problema que aborden. El segundo es que la parte del razonamiento no creativo que sin duda es necesaria para la generación de historias no tiene porque ser tan sencilla como la parte creativa, además de poder ser también, dependiente del dominio de conocimiento del problema.

5.3. Evaluación de historias

Con respecto a los métodos para evaluar las aplicaciones de narración automática desarrolladas con el armazón propuesto hay que distinguir dos propósitos distintos. El primero es la *evaluación interna* que consiste en probar que la hipótesis de este trabajo de investigación se cumple, tanto de una manera formal en los aspectos teóricos de esta propuesta, como de una manera empírica en los experimentos realizados en el laboratorio sobre la aplicación de ejemplo. El segundo propósito es la *evaluación externa* que consiste en verificar si el armazón propuesto funciona cuando es aplicado más allá de las fronteras del laboratorio, en el mundo real.

Teniendo esto presente, es fácil ver que los resultados de este trabajo se evaluaron en el sentido inverso al que habría sido más conveniente. En primer lugar, *ProtoPropp* fue desarrollado para ser evaluado ante un grupo de jueces humanos, con un método similar al utilizado por *Minstrel* o *Mexica*, diseñado para probar que las historias generadas son aceptadas por el público en general y comparables incluso con las obras creadas por autores humanos. A pesar de que los resultados, presentados en el apartado 4, son razonablemente satisfactorios y aportan algo de luz a las cuestiones básicas de la generación de historias, un análisis más profundo de este tipo de evaluación revela lo complejo, e inapropiado desde el punto de vista de la Informática, que resulta evaluar una aplicación de narración automática atendiendo a los criterios subjetivos del público y del propósito también subjetivo del material narrativo en sí mismo. Aunque el estilo lingüístico y la estructura de muchos sistemas es mejorable, como por ejemplo el caso de Minstrel, es evidente que más significativo que las historias concretas que se obtienen es la manera en que estas se generan. Por ello, en segundo lugar se propuso una evaluación más interna para la aplicación de ejemplo, de modo que pudieran realizarse pruebas formales automáticas que permitieran valorar con mayor rigor el funcionamiento de las aplicaciones y por tanto del armazón con el cual han sido desarrolladas. Al menos como punto de partida solido sobre el que edificar posteriores metodologías de evaluación que tomen en cuenta parámetros más externos relativos al contenido de las historias generadas, la forma en que el público las percibe, etc.

La evaluación de los generadores automáticos de historias es una cuestión

compleja porque no se conocen métodos formales con los que medir su calidad. El éxito o fracaso de una historia dependen no sólo de la habilidad del autor sino de la predisposición y las preferencias del público.

Para resolver de una forma práctica el problema de medir la calidad de una historia, existen propuestas como la de realizar una prueba similar al *Test de Turing* [Mat97] o establecer comparaciones estadísticas mediante la puntuación por jueces [CL02].

En el caso de este trabajo, las pruebas de evaluación externa realizadas sobre *ProtoPropp* consistieron en utilizar un grupo de evaluadores externos. Los evaluadores utilizan la aplicación y después rellenan un cuestionario en el que asignan una puntuación a cada aspecto de la aplicación que se quiere someter a evaluación, permitiendo obtener datos empíricos sobre la aceptación que tiene el sistema entre los usuarios.

En relación a las pruebas de evaluación interna realizadas sobre la aplicación de ejemplo se puede comentar que revelan datos más fiables de los que puede obtenerse mediante evaluación externa.

La capacidad de *Minstrel* de aprender automáticamente de sus creaciones es una propiedad "natural" de los sistemas CBR. Este tipo de sistemas no es muy usado para implementar aplicaciones de narración automática, ni tampoco es fácil encontrar aplicaciones que dispongan además de sistemas de aprendizaje automático.

Según Rafael Pérez y Pérez [PyPS04] *Brutus* carece de una verdadera evaluación del valor y la novedad de las historias que genera, *Minstrel* sólo evalúa la novedad y *Mexica* evalúa ambos factores.

El mayor problema de las aproximaciones clásicas como la de *Tale-Spin* o *Universe* es la utilización de reglas predeterminadas que dificultan el aprendizaje de nuevos mecanismos, mientras que como se dijo anteriormente en un sistema CBR el aprendizaje está incluido en su ciclo de funcionamiento.

Finalmente una característica importante del armazón y la aplicación de ejemplo desarrollados en este trabajo de tesis, es que ambos son de código y documentación libre, permitiendo su estudio y ampliación por parte de cualquier otro miembro de la comunidad científica. Esta característica no se encuentra prácticamente en ningún otro proyecto importante de narración automática, lo que complica bastante su revisión y su comparación con la propuesta aquí presentada.

5.4. Historias resultantes

En este apartado se discuten los resultados obtenidos en esta investigación, analizando su significado con respecto a los objetivos propuestos ini-

cialmente y comparándolos con los obtenidos por otros investigadores del área.

La evaluación de una aplicación de narración automática es una tarea que habitualmente realizan sus desarrolladores, dado que son los que establecen el propósito final de la misma y por tanto también los criterios para medir la satisfacción del público. El armazón simplemente trata de facilitar las herramientas a los desarrolladores para realizar dicha evaluación.

Concretamente, en el caso de la evaluación de *ProtoPropp*, distinguimos entre una evaluación automática, los resultados obtenidos con las métricas formales, y una evaluación empírica, los resultados obtenidos con los cuestionarios del experimento.

En el caso de la evaluación automática, debe tenerse en cuenta que los resultados son fuertemente dependientes de la forma en que estén configurados los pesos asignados a cada fórmula de cada tipo normalizado de cada componente, además del corpus de historias de la aplicación y la manera en que estén definidos los conceptos Valuable en cada componente. Estas mediciones se realizaron con la configuración "estándar" que se proporciona con los componentes básicos que se distribuyen con el armazón, y con la aplicación *ProtoPropp* "estándar". A posteriori es trivial realizar ajustes en las ontologías, el corpus y los pesos asignados a las fórmulas, de forma que el autor alcance cualquier resultados deseados, por eso era importante realizar estas mediciones sin haber hecho ningún experimento antes.

Los resultados relativos al valor de las historias son razonables, no existiendo diferencias muy significativas entre el 0.77 obtenido por la fábula del corpus y el 0.59 obtenido por la fábula generada. Probablemente esto es debido a que el corpus se creó una vez estaba establecida la ontología y la mayor parte de los conceptos Valuable, y la historia generada reproduce bastante fielmente el tipo de estructuras lógicas que pueden encontrarse en el corpus. El valor de la historia generada aleatoriamente sí es notablemente menor, como se esperaba.

En los resultados relativos a la novedad, el resultado de la historia aleatoria, 0.43, supera al de la historia generada mediante *ProtoPropp*, 0.33. Sin embargo la diferencia no es tan significativa como en un principio se esperaba, y de hecho ambos valores son bastante bajos, lo que sorprende especialmente en el caso de la historia aleatoria. Una posible explicación para este fenómeno es que los pesos en las fórmulas están demasiado cargados a nivel de la historia abstracta y la fábula, a pesar de que las estructuras de todos los cuentos a ese nivel tan alto son todas prácticamente iguales. Probablemente si se ignorase la similitud en ese nivel y se diese más importancia a las diferencias encontradas a más bajo nivel, como los tipos no normalizados de cada una de las funciones de personaje o el número de personajes por fábula,

etc. se obtendría una novedad mucho más alta para las historias. La historia del corpus existe en la base de conocimiento y por lo tanto no se encuentra diferencia alguna que aporte novedad.

En el caso de la evaluación empírica, la fábula recuperada del corpus actúa como punto de referencia para la evaluación. La historia generada es considerada por los jueces humanos como un cuento de valores aceptables, ya que las respuestas oscilan entre 4 y 6 en todas las preguntas del cuestionario.

Los resultados relativos a la coherencia son razonablemente predecibles y bastante favorables porque mientras que el cuento aleatorio consigue sólo un 1.81 los otros cuentos tienen medias más aceptables. El cuento generado obtiene los mejores resultados en coherencia, lo cual era uno de los objetivos más importantes del armazón: conseguir generar historias coherentes.

Algo similar ocurre con los resultados relativos al valor o interés de las historias. La fábula generada mediante *ProtoPropp* obtiene un aceptable 5.08, mientras que la aleatoria consigue un 3.23. Parece que el interés guarda algún tipo de relación con la coherencia, lo cual puede explicarse porque las personas no pueden apreciar el valor de una historia si esta les resulta incoherente, es decir, asumiendo que la coherencia es un requisito previo al valor de cualquier cosa.

La evaluación de la originalidad presenta alguna complicación que se pone de manifiesto en este experimento. Inicialmente se esperaba que la fábula aleatoria resultara más novedosa al diferenciarse claramente de otras historias dotadas de sentido común, pero el hecho es que la originalidad de la historia escogida directamente del corpus alcanza un valor de originalidad muy cercano, lo cual resulta extraño. En la evaluación de la misma propiedad, el generador de *ProtoPropp* fracasa, lo que también es en cierta medida un dato inesperado. La explicación para este fenómeno proviene seguramente de la forma en que el público interpreta la palabra "originalidad", entendiendo por esta la aparición de diversos individuos en la fábula y en la simulación, como ciertos personajes o agentes llamativos (como la rana), más que la estructura o configuración global de la historia abstracta. Aunque también existe una explicación alternativa y posiblemente complementaria, que dice que el "corpus mental" de cuentos que es capaz de recordar cada juez es diferente y por tanto, aunque estos aplicasen la misma medida de similitud entre la historia que se presenta al público y las que este conoce, los resultados en cuestión de novedad serían muy diferentes. Esto nos sugiere que la noción de novedad, mientras que no sea acotada de forma explícita, es relativa a la interpretación y el conocimiento de cada juez, siendo estas interpretaciones o conocimientos muy dispares entre sí. La desviación típica así lo muestra, diseminando los resultados en 2.5 puntos; ocurriendo algo similar en los comentarios informales recogidos durante el sondeo.

La cuarta propiedad, la calidad de la presentación lingüística, refleja la complejidad de evaluación de los textos en lenguaje natural, incluso aunque se traten de historias "básicas", muy similares en estructura a las historias abstractas que representan. La ejecución de un text ANOVA revela que la variación de la puntuación en calidad lingüística entre las tres historias es significativa a un nivel de significación del 5 %. Este resultado en principio resultaría chocante, ya que se ha utilizado el mismo método de generación de lenguaje basado en plantillas para los tres casos. Sin embargo el resultado se vuelve más comprensible al suponer que lo que influye de manera significativa en el público no es solo la estructura en sí del lenguaje sino también las palabras escogidas, en este caso en las plantillas, y el significado que resulta al verlas unidas, significado que el público no es capaz de separar del todo del significante, al menos en este experimento en particular. Estos resultados demuestran la conexión en un sentido entre la impresión que el público tiene del contenido de una historia (verdadero objeto de evaluación en un generador de fábulas) y la que tiene de su presentación. Aunque la conexión en el otro sentido no está demostrada, la hipótesis de que la presentación influye en la impresión de coherencia, valor y novedad que el público encuentra en una historia, parece cobrar más fuerza.

Para completar el análisis sobre los resultados de ProtoPropp, se cotejan los resultados obtenidos con las métricas formales con los obtenidos a través del experimento con jueces humanos. En cuando al valor formal, que equiparamos al interés manifestado por los jueces humanos, hay algunas diferencias importantes en los resultados. Concretamente la historia del corpus resulta más valiosa que la generada, según la métrica formal, mientras que según los jueces esto ocurre al revés. Hay dos motivos por los que estos resultados pueden haberse producido. El primero es que las historias del corpus fueron creadas tras tener fijados los conceptos Valuable de la ontología, y por lo tanto se elaboraron teniendo muy presentes los requisitos para que cada elemento fuese reconocido como valioso en la ontología. El segundo es que el público sí que no conoce en absoluto los criterios de valor establecidos en los componentes ontológicos del sistema, y siendo estos tan sencillos y escogidos *ad hoc*, es de esperar que no coincidan con los suyos. En cuanto a la novedad formal, que equiparamos a la originalidad manifestada por los jueces humanos, la mayor diferencia se produce en la historia del corpus. Mientras que la novedad formal de esta historia es cero, como no puede ser de otra formada dada la definición de esta métrica, el público le asigna un 5.04 de media sobre 10. Esta diferencia se debe a una interpretación distinta de la creatividad del sistema que se hacen en uno y otro caso, y en la que se profundiza más en el siguiente párrafo. Los resultados en la historias aleatoria y en la historia generada normalmente son razonablemente parecidos, teniendo

en cuenta la escala: 0.43 sobre 1 es prácticamente una décima menor a 5.44 sobre 10, y 0.33 también recae sólo unas centésimas por debajo de 3.92 sobre 10, lo que indica que el autor computacional es tan sólo un poco más severo que el público juzgando la novedad de sus creaciones. Como la coherencia es considerada un requisito según el formalismo de representación elegido, no es posible contrastar la coherencia formal de las historias con los resultados empíricos.

Según los términos de Boden[Bod90], los valores de creatividad que los jueces han asignado a las historias básicas deben entenderse en un sentido de la creatividad histórica, o h-creatividad, dado que los jueces no tienen acceso al corpus de la aplicación ni conocen otras historias generadas por *Proto-Propp*. La comparación se realiza, por tanto, entre las historias presentadas y todas las historias similares que cada juez es capaz de traer a la memoria en el momento de responder el cuestionario, probablemente los cuentos que conoce de toda la vida. Esta evaluación es aceptable, pero aún así no puede considerarse más que una aproximación informal a una verdadera medida de h-creatividad, dado que nadie en realidad conoce ni es capaz de tener presente al mismo tiempo todos los detalles de todas las historias de la Humanidad. Es por ello que la medida de creatividad computacional, c-creatividad, resulta más apropiada para una aplicación de narración automática, y esta debería obtenerse de forma interna al propio sistema informático, o en todo caso, facilitándole a los jueces el corpus de historias con las que deben comparar las historias que se les presentan antes de emitir su veredicto. Tras los experimentos, durante una charla informal con algunos de los jueces, se les informó de que la historia generada por *ProtoPropp* es en realidad muy similar al cuento número 131 de la colección de Afanasiev, un cuento clásico del folklore ruso que estaba incluido en el corpus de la aplicación. Esto significa que en realidad la novedad de la historia generada debería ser baja, aunque paradójicamente durante los experimentos la mayor parte de los jueces no reconocieron el cuento clásico y le asignaron una puntuación de novedad aceptable.

La evaluación de productos tan complejos como estas historias "básicas" se enfrenta a importantes problemas, generalmente provenientes de la naturaleza subjetiva de la finalidad para el que estos productos están hechos, a saber, satisfacer al público de la aplicación. Esto conlleva una necesidad lógica de contar con evaluadores humanos, posiblemente como complemento de otras medidas más cuantitativas sobre las historias, lo que complica el proceso de evaluación y la recopilación de datos fiables al respecto. Buena parte de esta complicación proviene del posible efecto "ruido" de la capa de presentación que afecta a la evaluación de los jueces, como es el caso de la generación basada en plantillas que presentamos aquí, a pesar de lo sencilla

y uniforme de su ejecución.

El trabajo teórico en la evaluación de sistemas creativos [Rit01] sugiere que hay dos magnitudes básicas que deben ser consideradas: el *valor* y la *tipicalidad*, medidas que se corresponden con el valor y la novedad en el sentido en que se utilizan en este trabajo. En realidad hablando en términos de la evaluación humana en vez de la evaluación máquina, y con respecto a los parámetros evaluados en este experimento, la segunda de estas correspondencias es la más natural ya que a los jueces se les insistió explícitamente en que cuando evaluaran tuviesen en cuenta que los resultados pretendían ser cuentos tradicionales, comparables sólo con los de ese tipo. La primera correspondencia, entre valores, es más problemática porque la evaluación del valor en un sentido amplio, y como ya mencionábamos antes, también debería incluir de alguna forma a la coherencia.

Hay otros importantes factores involucrados en el modo en que el público se enfrenta a una historia para su evaluación. Pérez y Pérez [PyPS01] realiza una proposición valiosa y muy detallada sobre una serie de factores. El sistema *Mexica* basa buena parte de su evaluación en la tensión que se crea en la historia y como esta evoluciona a lo largo de la misma, principalmente a través de los cambios en las relaciones emocionales que sufren los personajes, y las amenazas que su salud o su vida pueden llegar a sufrir. Las subidas y bajadas de tensión a lo largo de una historia son un factor clave sobre su valor, que no sería costoso de implementar en un componente específico que especializara *Narration*, dada la naturaleza declarativa de este tipo de información.

Aunque el tipo de ejecución de la aplicación de ejemplo no era un aspecto a considerar esta investigación, es posible comentar que los resultados muestran un rendimiento aceptable (menos de cinco minutos por historia) con las operaciones básicas relacionadas con el model de DLs, aunque el coste temporal se dispara en cuanto se añaden reglas SWRL al modelo que está cargado en el motor de razonamiento utilizado, por lo que no es posible establecer unas cotas precisas de duración para el proceso de generación de historia.

Capítulo 6

Conclusiones

Siempre que enseñes, enseña a la vez a dudar de lo que enseñas.
José Ortega y Gasset

Es necesario terminar este investigación con una reflexión sobre los resultados obtenidos que permita obtener conclusiones concretas acerca de lo que ha significado este trabajo. Dada la situación inicial de la investigación en narración automática, un tanto precaria en cuanto a consenso entre investigadores y comprensión e interés por parte de expertos de otras disciplinas, se pretende que este trabajo represente un paso adelante hacia esa "base científica rigurosa" que permitirá mejorar las condiciones en que se realiza la investigación en este campo.

En el capítulo 3 se propone un armazón para el desarrollo de aplicaciones de narración automática que ha sido construido para satisfacer tanto a los desarrolladores como a los usuarios finales del producto. Aunque este propósito era ambicioso por la dificultad que supone ponerlo práctica y evaluar objetivamente sus resultados, las primeras experiencias han sido positivas, de manera que es de esperar que se sigan recogiendo frutos de esta línea de investigación a medida que el armazón propuesto madure, aumente el número de desarrolladores que lo utilizan y el número de dominios y aplicaciones diferentes de narración automática sobre el que el método de desarrollo propuesto se aplica.

A continuación se presentan las conclusiones finales correspondientes a cada objetivo planteado en el apartado 1.4.

6.1. Genericidad y dependencia en autoría computacional

Tras un estudio razonablemente profundo y sólido del trabajo previo, se dispone de una teoría más clara y actualizada que explica qué son las aplicaciones de narración automática, cómo funcionan y cuales son los principales problemas a los que se enfrentan sus desarrolladores.

En base a ideas tomadas de diferentes aplicaciones reales se ha creado un armazón para implementar algoritmos de autoría computacional para la narración automática. Este armazón cumple con los objetivos propuestos al permitir la generación automática de nuevas fábulas a partir de una solicitud inicial y una base de conocimiento, la construcción automática de un discurso apropiado elaborando contenidos de ficción razonablemente genéricos y la presentación de la historia obtenida en forma independiente.

Como se planteó en un principio, este armazón admite parámetros para su configuración y es posible realizar experimentos automáticos para evaluar su funcionamiento.

Los algoritmos de generación y evaluación podrán ser integrados en arquitecturas modulares, altamente reutilizables y mantenibles. Aunque en el modelo de conocimiento se han incluido conceptos básicos sobre narración automática, dicho modelo puede extenderse o sustituirse por completo con conceptos de dominios más específicos, gracias al uso de componentes ontológicos reutilizables.

Sin embargo también existen carencias en el armazón propuesto, puestas de manifiesto en los resultados obtenidos. La cuestión de la novedad de las historias, al tratarse de c-creatividad y no de h-creatividad, está fuertemente ligada al contenido de la base de conocimiento. Esta evaluación merece una valoración distinta según sea su tamaño, la completitud y complejidad de su contenido y la diferencia que existe entre dicho contenido y las historias que se evalúan.

De entre las técnicas estudiadas, la combinación del razonamiento basado en casos con otras técnicas de creatividad computacional, abre las puertas a nuevas soluciones, pero también a nuevos errores de generación debido a recuperaciones o adaptaciones incorrectas, que pueden reducirse significativamente gracias a un adecuado modelado del conocimiento implícito que se halla en la base de casos.

Los componentes de dominio son también otro punto clave pero conflictivo del armazón, dado que haría falta elaborar un mayor número de ellos para poder probar realmente su genericidad, así como para conocer hasta donde puede llegar la complejidad de los mismos. Un ejemplo de componen-

te cuya complejidad podría enriquecer bastante la salida de las aplicaciones es *Discourse*, al que se le ha prestado menor atención que a *Fabula* porque su generación está en una etapa más alejada dentro del proceso propuesto de autoría computacional.

Los parámetros y los ficheros de configuración son algo diferentes en cada componente básico, lo que indica que todavía hay decisiones de diseño complejas y difíciles de estandarizar que deben dejarse al juicio del desarrollador de la aplicación.

La capa de presentación se ha mantenido independiente de la representación basada en DLs del armazón. El trabajo se limita a usar tecnología independiente para mostrar las historias al público, como ocurre con las plantillas y el texto enlatado en lenguaje natural utilizado para evaluar *ProtoPropp*.

6.2. Evaluación automática con métricas abstractas

El tercer objetivo propuesto inicialmente se ha alcanzado definiendo conjuntos de métricas asociados a cada componente del armazón. Esta métricas están ciertamente integradas en la aplicación final, aunque son claramente dependientes de los componentes que las implementan y operan en su mismo nivel de abstracción.

La evaluación de acuerdo a estas métricas puede ser fácilmente automatizada, gracias a la capacidad de inferencia de las DLs, resultando además muy eficiente, siendo esto una gran ventaja a la hora de llevar a cabo un gran número de pruebas. Los resultados se crean, clasifican y evalúan de manera completamente automática, ofreciendo claras oportunidades de enriquecer la base de conocimiento con las historias generadas. Este tipo de evaluación es una aproximación de la medición real de la satisfacción de los usuarios que obtendríamos con el producto final, presentando las historias mediante texto o imágenes.

La alternativa clásica consiste en utilizar jueces humanos para evaluar las historias generadas mediante cuestionarios donde se puntúan distintas características del resultado. Estas pruebas tienen el valor de proporcionar datos empíricos sobre el funcionamiento de las aplicaciones con usuarios reales; pero también se ven contaminadas por el "ruido", de proporción desconocida, que introduce la propia capa de presentación en el proceso de evaluación, además de la subjetividad a la que están sometidos este tipo de experimentos.

Se concluye que una adecuada combinación de estos dos tipos de evaluación es la solución más completa para realimentar el proceso de desarrollo de

aplicaciones de narración automática.

6.3. Validez delegada al modelo de evaluación

Por último, se ha construido un armazón, diseñado de acuerdo a los principios de la programación orientada a objetos y al diseño modular de ontologías, que integra los algoritmos de generación y las métricas de evaluación en un sólo producto.

El objetivo de esta investigación es proporcionar un armazón para el desarrollo de aplicaciones de narración automática que satisfaga tanto a los usuarios como a los desarrolladores, lo que se pone a prueba con la aplicación de ejemplo presentada en el capítulo 4. Esta aplicación, lejos de sustituir la creatividad humana, actúa como un potenciador del esfuerzo del autor, dado que aplica el poder lógico y combinatorio del ordenador para generar nuevas y valiosas historias que se apoyan sobre el conocimiento declarativo y operacional, siempre de origen humano, que constituyen la base de conocimiento.

Sobre esta aplicación se han realizado pruebas automáticas que permiten obtener datos razonablemente fiables de los que extraer conclusiones.

La conclusión que se deriva de los datos es positiva en el sentido de que podemos construir aplicaciones que, de forma bastante eficiente, generen historias satisfactorias de acuerdo a las métricas que definamos. Los límites serán aquellos establecidos en las métricas y el modelo de conocimiento de los componentes de dominio que se usan en las aplicaciones finales.

6.4. Trabajo futuro

Con la intención de completar las conclusiones, se presentan en este apartado algunas líneas de investigación prometedoras que puede servir para orientar el trabajo futuro en este área.

Por un lado es posible continuar con las tareas relacionadas con el mantenimiento, la corrección y la mejora del software y la documentación generados en este proyecto de investigación. Es importante considerar que el uso de *DLApplication* para desarrollar nuevas aplicaciones de narración automática permitirá que los conceptos de esta tesis maduren aún más, dando mayor peso práctico a la evaluación de este trabajo, por lo que son muy importantes. Los nuevos componentes de dominios tanto generales como específicos que se desarrollen también completarán el repositorio del armazón, haciendo aún más evidente su reusabilidad y utilidad práctica. Todos los tutoriales, instaladores, manuales y ejemplos de uso que se añadan también serán be-

neficiosos para que tanto el software como las ideas en que este se apoya se difundan más y beneficien a más desarrolladores e investigadores en el área.

Por otro lado, la exploración de las cuestiones de presentación multimodal de las historias y su narración interactiva son las líneas de investigación más prometedoras de cara a encontrar posibles aplicaciones industriales para este trabajo. Las conexiones con los generadores de lenguaje natural pueden ser mejoradas añadiendo hipertexto, imágenes y sonidos prefabricados a las historias generadas. Incluso las conexiones con entornos virtuales pueden ser consideradas para representar dramáticamente la historia usando modernos motores 3D o narrar de forma interactiva usando funcionalidades avanzadas de estos entornos como la simulación física realista, los personajes autónomos o semiautónomos, el tiempo real, etc.

El futuro desarrollo de herramientas de autoría para este armazón, un aspecto fuera del alcance de este trabajo, será decisivo para el éxito del mismo en un contexto más práctico. El desarrollo a nivel industrial de aplicaciones de narración automática debe delegar la creación de los recursos y la base de conocimientos inicial a los expertos en los dominios concretos de la aplicación (guionistas, pedagogos, artistas, etc.). No hay razón para que estos expertos deban saber usar sofisticadas herramientas de ingeniería de conocimiento y programación, siendo más razonable facilitarles el uso herramientas de autoría de más alto nivel para realizar las tareas específicas de su dominio.

Las técnicas de modelado de usuario no han sido abordadas en este estudio, pero son otras posibles herramientas para ajustar los resultados a los criterios de valor y novedad subjetivos de cada usuario. La programación con restricciones también ha sido sugerida como una posible integración entre generación y evaluación de historias abstractas en un mismo enfoque de resolución del problema.

Aunque animados por los progresos en cuestiones de autoría computacional, cada vez más notorios, el hecho de que la inteligencia y la creatividad artificiales sigan siendo objeto de profunda investigación no significa que el desarrollo de aplicaciones de narración automática deba permanecer desatendido por el momento; en realidad, incluso bajo el supuesto de que no fuese posible replicar a la perfección el talento humano, el desarrollo de aplicaciones de narración automática está justificado. Como Janet Murray [Mur97] nos recuerda, la repetición de patrones y fórmulas ha existido desde siempre en el Arte y la Cultura, realizando una función nada despreciable que complementa y saca provecho de las grandes creaciones revolucionarias. Esta función es precisamente la de establecer las convenciones propias de una disciplina en un contexto histórico determinado, fijando de forma implícita los límites, más exigentes cada vez, que el ingenio humano debe superar para poder ser reconocido como tal.

Referencias

[Apaa] Apache. Commons logging. `http://commons.apache.org/logging/` [3/2/2008].

[Apab] Apache. log4j. `http://logging.apache.org/log4j/` [3/2/2008].

[Ari74] Aristóteles. *Poética*, volume 8 of *Coleccion Biblioteca Románica Hispánica IV*. Gredos, D.L., Madrid, 1974.

[Bal98] Mieke Bal. *Narratology: Introduction to the Theory of Narrative*. University of Toronto Press, second edition, 1998.

[Bat] John A Bateman. John bateman's ontology portal. `http://www.fb10.uni-bremen.de/anglistik/langpro/webspace/jb/info-pages/ontology/ontology-root.htm` [3/2/2008].

[Bat92] Joseph Bates. Virtual reality, art and entertainment. *PRESENCE: Teleoperators and Virtual Environments*, 1(1):133–138, 1992.

[BCM+03] Franz Baader, Diego Calvanese, Deborah L McGuinness, Daniele Nardi, and Peter F Patel-Schneider. *The Description Logic Handbook. Theory, Implementation and Applications*. Cambridge University Press, UK, 2003.

[Bec] Dave Beckett. Redland rdf libraries. `http://librdf.org/` [3/2/2008].

[BF99] Selmer Bringsjord and David A Ferrucci. *Artificial Intelligence and Literary Creativity: Inside the mind of Brutus, a storytelling machine*. Lawrence Erlbaum Associates, Hillsdale, NJ, first edition, 1999.

[BHR95] John A Bateman, Renate Henschel, and Fabio Rinaldi. The generalized upper model 2.0. Technical report, 19/12 1995.

[BMC03] Sean Bechhofer, Ralf Moller, and Peter Crowther. The DIG description logic interface. In *Description Logics*. CEUR Workshop Proceedings, 2003.

[Bod90] Margaret A Boden. *The Creative Mind: Myths and Mechanisms*. Wedenfield and Nicholson Ltd, London, 1990.

[BvHH+] Sean Bechhofer, Frank van Harmelen, Jim Hendler, Ian Horrocks, Deborah L McGuinness, Peter F Patel-Schneider, and Andrea Stein. OWL web ontology language reference. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/` [3/2/2008].

[Cam68] Joseph Campbell. *The Hero with a Thousand Faces*. Bollingen series. Princeton University Press, Princeton, second edition, 1968.

[CCPL] Marc Cavazza, Fred Charles, David Pizzi, and Jean Luc Lugrin. Intelligent virtual environments projects. `http://ive.scm.tees.ac.uk/` [3/2/2008].

[CDF+] Monica Crubézy, Olivier Dameron, Ray W Fergerson, Holger Knublauch, Mark A Musen, Natasha F Noy, Daniel Rubin, Samson W Tu, and Jennifer Vendetti. Protégé project. `http://protege.stanford.edu/` [3/2/2008].

[Cha86] Seymour Chatman. *Story and Discourse : Narrative Structure in Fiction and Film*. Cornell University Press, USA, second edition, 1986.

[CHI] CHISEL. Jambalaya. `http://www.thechiselgroup.org/jambalaya` [3/2/2008].

[CL02] Charles B. Callaway and J. C Lester. Narrative prose generation. *Artificial Intelligence*, 139(2):213–252, 2002.

[CO] CO-ODE. OWLdoc. `http://www.co-ode.org/downloads/owldoc/` [3/2/2008].

[Cra] Chris Crawford. Erasmatron & erasmaganza. `http://www.erasmatazz.com/` [3/2/2008].

[Cra05] Chris Crawford. *On Interactive Storytelling*. New Riders: Game Design and Development. Peachpit Press, Berkeley, California, first edition, 2005.

[Cul81] R. E Cullingford. Sam. In Roger C Schank and Christopher K Riesbeck, editors, *Inside Computer Understanding*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

[Cyc] Cycorp. The cyc knowledge base. `http://www.cyc.com/` [3/2/2008].

[DAGC03] Belén Díaz-Agudo and Pedro Antonio González-Calero. Knowledge intensive CBR through ontologies. *Expert Update*, 6(1), 2003.

[DAGP04] Belén Díaz-Agudo, Pablo Gervás, and Federico Peinado. A case based reasoning approach to story plot generation. In *Advances in Case Based Reasoning. Proceedings of the 7th European Conference on Case Based Reasoning*, volume 3155 of *Lecture Notes in Artificial Intelligence*, pages 142–156, Madrid, Spain, 2004. Springer Verlag.

[DARa] DARPA. Daml+oil. `http://www.daml.org/2001/03/daml+oil-index.html` [3/2/2008].

[DARb] DARPA. The darpa agent markup language homepage. `http://www.daml.org/` [3/2/2008].

[DMV02] Steffi Domike, Michael Mateas, and Paul Vanouse. A recombinant history apparatus presents: Terminal time. In *Narrative Intelligence*. John Benjamins Press, 2002.

[Doy] Patrick Doyle. The virtual theater proyect. `http://ksl-web.stanford.edu/projects/cait/` [3/2/2008].

[DSS93] Randall Davis, Howard Shrobe, and Peter Szolovits. What is a knowledge representation? *AI Magazine*, 14(1):17–33, 1993.

[Ecl] Eclipse.org. Eclipse. `http://www.eclipse.org/` [4/2/2008].

[eDr] eDrama. edrama learning. `http://www.edrama.co.uk/` [4/2/2008].

[Ext] Extempo. Extempo systems inc. `http://www.extempo.com` [4/2/2008].

[Fie98] Syd Field. *Screenplay: The Foundations of Screenwriting*. Amazon.com, third edition, 1998.

[For41] E. M Forster. *Aspects of the Novel*. Edward Arnold, London, 1941.

[GAGCDAFC99] Mercedes Gómez-Albarrán, Pedro Antonio González-Calero, Belén Díaz-Agudo, and Carlos Fernández-Conde. Modelling the CBR life cycle using description logics. In *International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development*, volume 1650 of *Lecture Notes in Artificial Intelligence*, pages 147–161, Seeon Monastery, Germany, 1999. Springer Verlag.

[GBPMRI07] Marina Gallego Barrigón, Irene Pérez Medina, and Almudena Ruiz Iniesta. Generación de cuentos interactivos usando CBR. Memoria de proyecto de sistemas informáticos, Facultad de Informática, Universidad Complutense de Madrid, 12/7 2007.

[GCDABT$^+$] Pedro Antonio González-Calero, Belén Díaz-Agudo, Juan José Bello-Tomás, Juan Antonio Recio-García, Antonio A Sanchez-Ruiz Granados, Derek Bridge, Lisa Cummins, Josep Lluís Arcos, Nirmalie Wiratunga, and Sutanu Chakraborti. jCOLIBRI CBR framework. `http://gaia.fdi.ucm.es/projects/jcolibri/` [4/2/2008].

[GDAPH04] Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás. Story plot generation based on CBR. In *Annual International Conference of the British Computer Society's Specialist Group on Artificial Intelligence (SGAI). Applications and Innovations in Intelligent Systems XII*, WICS, pages 33–46, Cambridge, UK, /12 2004. Springer Verlag.

[GDAPH05] Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás. Story plot generation based on CBR. *Knowledge-Based Systems. Special issue: AI-2004*, 18(4-5):235–242, 2005.

[GI03] Andrew S Gordon and Nicholas V Iuppa. Experience management using storyline adaptation strategies. In *International Conference on Technologies for Digital Storytelling and Entertainment*, Darmstadt, Germany, 24-26/3 2003.

[GLRMP06] Pablo Gervás, Birte Lönneker-Rodman, Jan Christoph Meister, and Federico Peinado. Narrative models: Narratology meets artificial intelligence. In *International Conference on Language Resources and Evaluation. Satellite Workshop: Toward Computational Models of Literary Analysis*, pages 44–51, Genova, Italy, 22/5 2006.

[GMF+02] J Gennari, Mark A Musen, Ray W Fergerson, W E Grosso, Monica Crubézy, H Eriksson, Natasha F Noy, and Samson W Tu. The evolution of Protégé: An environment for knowledge-based systems development. Technical report, Stanford University, 2002.

[GNU] GNU. Gnu lesser general public license. `http://www.gnu.org/copyleft/lesser.html` [4/2/2008].

[Gru93] T R Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993.

[Ham86] K. J. Hammond. Chef: A model of case-based planning. In *National Conference on Artificial Intelligence*, Philadelphia, PA, Fifth 1986.

[Her] David Herman. Narratology as a cognitive science. `http://www.imageandnarrative.be/narratology/davidherman.htm` [4/2/2008].

[Hir98] Haym Hirsh. Trends & controversies: Interactive fiction. *IEEE Intelligent Systems*, 13(6):12–21, 1998.

[HM03] Volker Haarslev and Ralf Möller. Racer user's guide and reference manual version 1.7.7. Technical report, Concordia University and University of Applied Sciences in Wedel, /11 2003.

[HP] Hewlett-Packard. Jena: A semantic web framework for java. `http://jena.sourceforge.net/` [4/2/2008].

[HRW+96] Larry F Hodges, Barbara O Rothbaum, Benjamin A Watson, G. Drew Kessler, and Dan Opdyke. Virtual reality exposure for fear of flying therapy. *IEEE Computer Graphics Applications*, 16(6):42–49, 1996.

[Hug88] John Hughes. Therapy is fantasy: Roleplaying, healing and the construction of symbolic order. In *Anthropology IV Honours, Medical Anthropology Seminar*, Dept. of Prehistory & Anthropology, Australian National University, 1988.

[IH] International-Hobo. Foundations of interactive storytelling. `http://www.igda.org/writing/InteractiveStorytelling.htm` [4/2/2008].

[ISO] ISO/IEC. Jtc1/sc22/wg21 - the C++ standards committee. `http://www.open-std.org/JTC1/SC22/WG21/` [4/2/2008].

[KAB+73] Sheldom Klein, John F Aeschlimann, David F Balsiger, Steven L Converse, Claudine Court, Mark Forster, Robin Lao, John D Oakley, and Joel Smith. Automatic novel writing: A status report. Technical Report 186, Computer Science Department, University of Wisconsin, 1973.

[KL96] Janet L Kolodner and David Leake. A tutorial introduction to case-based reasoning. In *Case-Based Reasoning: Experiences, Lessons and Future Directions*, pages 31–65. MIT Press, 1996.

[KLO86] A. M Kass, D. B Leake, and C Owens. Swale: A program that explains. In Roger C Schank, editor, *Explanation Patterns: Understanding Mechanically and Creatively*, pages Appendix 232–254. Lawrence Erlbaum Associates, Hillsdale, NJ, 1986.

[Kol83] Janet L Kolodner. Reconstructive memory, a computer model. *Cognitive Science*, 7:281–328, 1983.

[Kol84] Janet L Kolodner. *Retrieval and Organizational Strategies in Conceptual Memory*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1984.

[Kol93] Janet L Kolodner. Understanding creativity: A case-based approach. In *European Workshop on Case-Based Reasoning*, Kaiserslautern, Germany, 1993. Springer Verlag.

[KP90] Janet L Kolodner and T L Penberthy. A case-based approach to creativity in problem solving. In *Annual Conference of the Cognitive Science Society*, Cambridge, MA, /8 1990.

[LÖ4] Birte Lönneker. Lexical databases as resources for linguistic creativity: Focus on metaphor. In *International Conference on Language Resources and Evaluation: Workshop on Language Resources for Linguistic Creativity*, pages 9–16, Lisbon, Portugal, 2004. European Language Resources Association.

[Lak87] George P Lakoff. *Women, Fire and Dangerous Things: What Categories Reveal about the Mind*. The University of Chicago Press, Chicago, USA, 1987.

[Lan97] R. R Lang. *A Formal Model for Simple Narratives*. PhD thesis, Tulane University, 1997.

[Lau91] Brenda Kay Laurel. *Computers as Theatre*. Addison-Wessley Publishing, New York, 1991.

[Leb87] Michael Lebowitz. Storytelling and generalization. In *Annual Conference of the Cognitive Science Society*, pages 100–109, Berkeley, California, 1987.

[Lee94] M Lee. *A Model for Story Generation. M.Sc. Thesis*. PhD thesis, University of Manchester, 1994.

[LS02] Hugo Liu and Push Singh. Makebelieve: Using commonsense to generate stories. In *National Conference on Artificial Intelligence*, pages 957–958, Edmonton, Alberta (Canada), 28/7-1/8 2002. AAAI Press.

[LSE] Hugo Liu, Push Singh, and Ian Eslick. The conceptnet project. `http://web.media.mit.edu/\~hugo/conceptnet/` [4/2/2008].

[Lyt92] Steven L Lytinen. Conceptual dependency and its descendants. *Computers and Mathematics with Applications*, 23(2-5):51–73, 1992.

[LZ02] Ruqian Lu and Songmao Zhang. *Automatic generation of computer animation: Using AI for movie animation*, volume 2160 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, Berlin-Heidelberg-New York, 2002.

[Mag02] Brian Magerko. A proposal for an interactive drama architecture. In *AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, Stanford, CA, 2002. AAAI Press.

[Mat97] Michael Mateas. An oz-centric review of interactive drama and believable agents. *AI Today: Recent Trends and Developments. Lecture Notes in Artificial Intelligence*, 1600:297–328, 1997.

[MBI03] William J Mitchell, Marjory S Blumenthal, and Alan S Inouye. *Beyond Productivity: Information, Technology, Innovation, and Creativity*. National Academies Press, 2003.

[McK97] Robert McKee. *Story. Substance, Structure, Style and the Principles of Screenwriting*. Regan Books, New York, 1997.

[Med] DG Information Society and Media. Ict work programme 2007-08. `ftp://ftp.cordis.lu/pub/fp7/ict/docs/ict-wp-2007-08\_en.pdf` [18/2/2008].

[Mee76] James Richard Meehan. *The Metanovel: Writing Stories by Computer*. PhD thesis, 1976.

[Mee81] James Richard Meehan. Tale-spin and micro tale-spin. In Roger C Schank and Christopher K Riesbeck, editors, *Inside computer understanding*. Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

[Mei] Jan Christoph Meister. Narrnet: Narratology network. `http://www.narratology.net/` [18/2/2008].

[Men] Object Mentor. Junit. `http://www.junit.org/` [18/2/2008].

[Mina] Mindswap. Pellet OWL reasoner. `http://pellet.owldl.com/` [18/2/2008].

[Minb] Mindswap. Swoop: A hypermedia-based featherweight OWL ontology editor. `http://code.google.com/p/swoop/` [18/2/2008].

[MIT01] MIT. *The MIT Encyclopedia of the Cognitive Sciences*. Bradford Books, 2001.

[MOOK98] C Mellish, M O'Donnell, J Oberlander, and A Knott. An architecture for opportunistic text generation. In *International Workshop on Natural Language Generation*, Niagra, 1998.

[MS99] Michael Mateas and Phoebe Sengers. Narrative intelligence: An introduction to the ni symposium. In *Working Notes of the Narrative Intelligence Symposium, AAAI Fall Symposium Series.*, pages 1–10, Menlo Park, California, /11 1999. AAAI Press.

[MS03] Michael Mateas and Andrew Stern. Façade: An experiment in building a fully-realized interactive drama. In *Game Developer's Conference, Game Design track*, San Jose, California, USA, /3 2003.

[Mue87] Erik T Mueller. Daydreaming and computation: A computer model of everyday creativity, learning, and emotions in the human stream of thought. doctoral dissertation. Technical Report UCLA-AI-87-8, Computer Science Department, University of California, 1987.

[Mur97] Janet Horowitz Murray. *Hamlet on the Holodeck. The Future of Narrative in Cyberspace*. MIT Press, Cambridge, MA, 1997.

[NI07] Arturo Nakasone and Mitsuru Ishizuka. ISRST: An interest based storytelling model using rhetorical relations. In *Edutainment*, volume 4469 of *Lecture Notes in Computer Science*, pages 324–335, Hong Kong, China, 11-13/6 2007. Springer.

[Nir] Sergei Nirenburg. Mikrokosmos ontology. `http://crl.nmsu.edu/Research/Projects/mikro/index.html` [18/2/2008].

[OTK] On-To-Knowledge. Oil description. `http://www.ontoknowledge.org/oil/` [18/2/2008].

[Peia] Federico Peinado. DLApplication, a framework for dealing with description logics. `http://federicopeinado.com/projects/dlapplication/` [18/2/2008].

[Peib] Federico Peinado. DLModel, a tool for dealing with description logics. `http://federicopeinado.com/projects/dlmodel/` [18/2/2008].

[Pei04] Federico Peinado. Mediación inteligente entre autores e interactores para sistemas de narración digital interactiva. Technical report, Departamento de Sistemas Informáticos y Programacion. Facultad de Informática. Universidad Complutense de Madrid, /9 2004.

[PG05a] Federico Peinado and Pablo Gervás. Creativity issues in plot generation. In *Workshop on Computational Creativity, Working Notes. 19th International Joint Conference on Artificial Intelligence*, pages 45–52, Edinburgh, UK, 30/7 - 5/8 2005. Technical Report 5-05. Depto. Sistemas Inform ticos y Programación, Universidad Complutense de Madrid.

[PG05b] Federico Peinado and Pablo Gervás. A generative and case-based implementation of proppian morphology. In Birte Lönneker, Jan Christoph Meister, Pablo Gervás, Federico Peinado, and Michael Mateas, editors, *Joint International Conference of the Association for Computers and the Humanities and the Association for Literary and Linguistic Computing*, Story Generators: Models and Approaches for the Generation of Literary Artifacts, pages 129–131, Victoria, Canada, 15-18/6 2005. University of Victoria.

[PG06a] Federico Peinado and Pablo Gervás. Evaluation of automatic generation of basic stories. *New Generation Computing, Computational Paradigms and Computational Intelligence. Special issue: Computational Creativity*, 24(3):289–302, 2006.

[PG06b] Federico Peinado and Pablo Gervás. Minstrel reloaded: From the magic of lisp to the formal semantics of OWL.

In *International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, volume 4326 of *Lecture Notes in Computer Science*, pages 93–97, Darmstadt, Germany, 4-6/12 2006. Springer.

[PGDA04] Federico Peinado, Pablo Gervás, and Belén Díaz-Agudo. A description logic ontology for fairy tale generation. In *International Conference on Language Resources and Evaluation: Workshop on Language Resources for Linguistic Creativity*, pages 56–61, Lisbon, Portugal, 29/5 2004. European Language Resources Association.

[PGDAH] Federico Peinado, Pablo Gervás, Belén Díaz-Agudo, and Raquel Hervás. Protopropp, the fairy-tale generator. `http://federicopeinado.com/projects/kiids/apps/protopropp/` [18/2/2008].

[pro] LaTeX project. LaTeX project: LaTeX - a document preparation system. `http://www.latex-project.org/` [18/2/2008].

[Pro68] Vladimir Propp. *Morphology of the Folktale*. University of Texas Press, USA, Austin and London, 1968.

[PyPS01] Rafael Pérez y Pérez and Mike Sharples. Mexica: a computer model of a cognitive account of creative writing. *Journal of Experimental and Theoretical Artificial Intelligence*, 13(2):119–139, 2001.

[PyPS04] Rafael Pérez y Pérez and Mike Sharples. Three computer-based models of storytelling: Brutus, minstrel and mexica. *International Journal on Knowledge Based Systems*, 17:15–29, 2004.

[RAE] RAE. Diccionario de la lengua española. vigésimo segunda edición. `http://www.rae.es` [18/2/2008].

[Rec03] Alan L Rector. Modularisation of domain ontologies implemented in description logics and related formalisms including OWL. In *K-CAP*, Sanibel Island, Florida, USA, 23Ű25/10 2003. ACM.

[RGDAGCS06] Juan A. Recio-García, Belén Díaz-Agudo, Pedro A. González-Calero, and Antonio Sánchez. Ontology based

CBR with jCOLIBRI. applications and innovations in intelligent systems xiv. In *SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, Cambridge, United Kingdom, /12 2006. Springer.

[RHK+95] B. O Rothbaum, L. F Hodges, R Kooper, D Opdyke, J Williford, and M. M North. Effectiveness of computer-generated (virtual reality) graded exposure in the treatment of acrophobia. *American Journal of Psychiatry*, 152(4):626–628, 1995.

[Rit01] Graeme D Ritchie. Assessing creativity. In *AISB Symposium on Articifial Intelligence and Creativity in Arts and Sciences*, pages 3–11, 2001.

[RS] Racer-Systems. RacerPro. `http://www.racer-systems.com/` [18/2/2008].

[RS89] Christopher K Riesbeck and Roger C Schank. *Inside Case-Based Reasoning*. Lawrence Erlbaum Associates, Evanston, Illinois, 1989.

[Rum75] David E Rumelhart. Notes on a schema for stories. In Daniel G Bobrow and Allan Collins, editors, *Representation and Understanding: Studies in Cognitive Science*, pages 211–236. Academic Press, Inc, New York, 1975.

[RY04] Mark O Riedl and R Michael Young. A planning approach to story generation for history education. In *International Conference on Narrative and Interactive Learning Environments*, 2004.

[Sch69] Roger C Schank. *A conceptual dependency representation for a computer-oriented semantics*. Phd thesis, University of Texas, 1969.

[Sch82] Roger C Schank. *Dynamic Memory*. Cambridge University Press, 1982.

[SD94] W Sack and M Davis. IDIC: Assembling video sequences from story plans and content annotations. In *Institute of Electrical and Electronics Engineers International Conference on Multimedia Computing and?Systems*, Boston, MA, 14-19/3 1994.

[Sgo99] Nikitas M Sgouros. Dynamic generation, management and resolution of interactive plots. *Artificial Intelligence*, 1(107):29–62, 1999.

[SJMM07] James Skorupski, Lakshmi Jayapalan, Sheena Marquez, and Michael Mateas. Wide ruled: A friendly interface to author-goal based story generation. In *Virtual Storytelling: Using Virtual Reality Technologies for Storytelling*, volume 4871 of *Lecture Notes in Computer Science*, pages 26–37, Saint-Malo, France, 5-7/12 2007. Springer.

[SLTW] Lewis Seifert, Celeste Lim, Laura Tan, and Nicole Wee. Digital propp: Proppian fairy tale generator. `http://www.brown.edu/Courses/FR0133/Fairytale\_Generator/index.html` [18/2/2008].

[SM] Sun-Microsystems. Java products & technologies. `http://www.java.sun.com/` [18/2/2008].

[Sun] Sun. Javadoc tool home page. `http://java.sun.com/j2se/javadoc/` [18/2/2008].

[Szi99] Nicholas Szilas. Interactive drama on computer: Beyond linear narrative. In *AAAI Fall Syposium on Narrative Intelligence*, pages 150–156, 1999.

[Tig] Tigris. ArgoUML. `http://argouml.tigris.org/` [8/1/2008].

[Tur92] Scott R Turner. Minstrel: A computer model of creativity and storytelling. Technical Report UCLA-AI-92-04, Computer Science Department, University of California, 1992.

[Tur98] Mark Turner. *The Literary Mind: The Origins of Thought and Language*. Oxford University Press, UK, 1998.

[W3Ca] W3C. Resource description framework (rdf). `http://www.w3.org/RDF/` [19/2/2008].

[W3Cb] W3C. Swrl: A semantic web rule language combining OWL and RuleML. `http://www.w3.org/Submission/SWRL/` [18/2/2008].

[Wil81] R Wilensky. Pam. In Roger C Schank and Christopher K Riesbeck, editors, *Inside Computer Understanding: Five Programs Plus Miniatures*, pages 136–179. Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

[Yaz89] M Yazdani. *Computational story writing*. Computers and Writing. Intellect books, Oxford, UK, 1989.

[ZMFV+05] José P Zagal, Michael Mateas, Clara Fernández-Vara, Brian Hochhalter, and Nolan Llchtl. Towards an ontological language for game analysis. In Suzanne deCastell and Jennifer Jenson, editors, *International Conference of the Digital Games Research Association*, pages 3–14, Vancouver, Canada, 2005. DIGRA.

[Zoe] Zoesis. Zoesis studios. `http://www.zoesis.com` [19/2/2008].

Apéndice A

Documentación complementaria

En este apéndice se presenta una lista de toda la documentación complementaria de esta tesis que se encuentra disponible en soporte electrónico para todos aquellos lectores que deseen conocer más detalles sobre esta investigación.

1. Versión en formato PDF de esta memoria de tesis doctoral.

2. Versión en formato PDF de todas las publicaciones relacionadas.

3. Última versión estable de *jDLModel*, la implementación *Java* del API *DLModel*, ejemplos y documentación *JavaDoc*.

   `http://federicopeinado.com/projects/dlmodel/`

4. Última versión estable de *jDLApplication*, la implementación *Java* del núcleo software *DLApplication*, con su código fuente, ejemplos y documentación *JavaDoc*.

   `http://federicopeinado.com/projects/dlapplication/`

5. Última versión estable de la implementación *Java* de la aplicación *ProtoPropp*, con su código fuente, ejemplos ejecutables, componentes ontológicos tanto básicos como específicos de esta aplicación, base de conocimiento y documentación *JavaDoc*.

   `http://federicopeinado.com/projects/kiids/apps/protopropp/`

Si desea recibir esta documentación complementaria o enviar alguna consulta o sugerencia al autor relacionada con esta tesis doctoral, puede hacerlo desde el sitio web del autor: `http://federicopeinado.com`

Apéndice B

English Translation

In order to satisfy the requirements of the application for the mention "Doctor Europeus" this appendix includes the English translation of the abstract and conclusions of this thesis, titled "A Framework for the Development of Automatic Storytelling Applications Based on Reusable Ontological Components".

Abstract

Computers have been used as a medium for narrative expression since decades. Computer Science has developed powerful technologies for narrative information processing, allowing us to store, organize, modify and reproduce every kind of content. Automatization of these task is definetly possible, but others has a extraordinary complex computational formalization, sometimes impossible, because the mental processes that the human being performs for working on those tasks are still unknown. Story generation is one of those tasks and its automatization is part of that big long-term project which is called "Artificial Intelligent".

Between seventies and eighties many studies were developed about story automatic comprehension and generation that establish the ground for scientific research in this topic. After year of lack of interest on this question, new applications, approaches and favourable technical conditions have prompted a short revival of interest on automatic storytelling applications in scientific and commercial contexts.

In this work theoretical bases of automatic storytelling are studied, reviewing different methodologies and technologies involved in its development and analyzing up-to-date results documented in the scientific literature.

Nowadays there are applications with the ability of generating content,

structure and final presentation of a story (using texts, graphics, videos, etc.) automatically. However these applications do not reveal a single and systematic solution for every fundamental problem in story generation. Although there are exceptions, many of these works fail in their scientific approach or achieve results difficult to generalize, scale, evaluate and use.

This work of thesis tries to establishing a more solid and suitable framework for the development of automatic storytelling applications. Goals are created about those problems that affect more seriously to the organization, modularization and reusability of two basic elements: on one hand, the knowledge needed to represent stories in the computer, on the other hand, processes able to manipulate that knowledge in order to generate stories that will satisfy preestablished criteria about value and novelty. The proposal uses description logics and object-oriented programming in order to build a framework composed by a software core and an extensible repository of ontological components that encapsulate declarative and operational semantics of key domains related to automatic storytelling.

Results are the implementation of this framework and an example application as its specific instance, that shows the most technical aspects about how to implement the proposed methodology for the development of automatic storytelling applications.

Evaluation of this proposal and its results are discussed, in terms of expressivity of this solution among others revised previously. It is also discussed what it is known about computational narrative authorship, the way stories are presented and evaluated, and the scope of the validation of that evaluation.

Finally conclusions of this research are presented, including some final suggestions about what should be the future work.

Keywords

Story Generation, Computational Narratology, Ontological Engineering, Description Logics, Computational Creativity

Conclusions

It is necessary to finish this research thinking about the achieved results in order to obtain concrete conclusions on what is the meaning of this work.

Given the current situation of the research on automatic storytelling, somehow precarious about consensus between researchers and experts from other disciplinesúnderstanding and interest, this work tries to represent a

step towards that "rigorous scientific base" that will allow improvements in the research conditions on this field.

In the chapter 3 a framework for the development of automatic storytelling applications has been proposed, specially designed to satisfy developers and final users of the product. Although this purpose was ambitious because of the difficulty in practical application and objective evaluation of results, first experiences are positive, so it is expected that more benefit will be produced thanks to this line of research, while the proposed framework matures, number of developers using it and domains and different automatic storytelling applications developed by this method rises.

Following final conclusions about each goal proposed in section 1.4 are presented.

Genericity and dependency in computational authorship

After a reasonably deep and solid study of previous work, a clearer and more updated that explains what are the interactive storytelling applications, how they work and what are the main problems their developers must confront.

Based on ideas taken from different real applications, a new framework for computational authorship algorithms has been created for automatic storytelling. This framework achieves the proposed goals allowing the automatic generation of new fabulas from an initial query and a knowledge base, the automatic construction of a suitable discourse producing reasonably generic fictional contents and the presentation of the final story in an independent form.

As it was initially proposed, this framework receives configuration parameters and it is possible to evaluate its functionality using automatic experiments.

Generation and evaluation algorithms can be integrated in modular, highly reusable and maintainable architectures. Although there are basic concepts about automatic storytelling included in the knowledge model, that model can be extended or completely substituted by concepts of more specific domains, thanks to the use of reusable ontological components.

However there are also lacks in the proposed framework, shown in the obtained results. The question of storiesñovelty, dealing with c-creatividad and not with h-creatividad, is strongly linked to the content of the knowledge base. This evaluation deserves a different valuation according to the size, completeness and complexity of the knowledge base, and the difference between it and the stories that are evaluated.

Among the reviewed techniques, combination of case based reasoning with other techniques of computational creativity opens the door to new solutions, but also to new possible generation failures due to wrong retrievals or adaptations, that can be significantly reduced thanks to a good modeling of the implicit knowledge found in the case base.

The domain components are also a key but conflictive point of the framework, because more of them should be necessary to really test their genericity in order to know how far its complexity can go. An example of a component which complexity can enrich the output of the applications is *Discourse*, what has been less developed than *Fabula* because it is used in a later stage of the proposed process of computational authorship.

Parameters and configuration files are slightly different in each basic component, what means that there are still complex and difficult to standardize design decision that must be taken by the developers of the final application.

The presentation layer is maintained independent from the DL-based representation of the framework has been a success. This work is limited to the use of independent technology for generating the presentation that is shown to the public, as the templates and canned text in natural language used for evaluating *ProtoPropp*.

Automatic evaluation with abstract metrics

Third goal proposed at the beginning of this work has been reached defining set of metrics associated with each component of the framework. These metrics are truly integrated in the final application, being clearly dependent on the components that implement them and work at the same abstraction level.

Evaluation according to these metrics can be easily automatized, thanks to the inference capacity of DLs, beings also very efficient, what is a great advantage for performing a large number of tests. Results are created, classified and evaluated in a completely automatic way, offering clear opportunities for enriching the knowledge base with generated stories. This kind of evaluation is an approach to the real assessment of users'satisfaction obtained from the final product, after presenting the stories using texts or images.

The classic alternative consists on using human judges for evaluating generated stories by filling questionnaires with scores for the results'characteristics. These tests have the value of providing empirical data about how applications work with real users; but they are also corrupted by the "noise", of unknown size, that the presentation layer itself introduce in the evaluation process, apart from the subjectivity that affects these experiments.

As a conclusions, a good combination of these two types of evaluation is

the most complete solution for providing feedback to the development process of automatic storytelling applications.

Validity delegated to the evaluation model

Finally, a framework has been built, designed according to object-oriented programming and modular design of ontologies principles, and integrated with the generation algorithms and the evaluation metrics in the same product.

The goal of this research is to provide a framework for the development of automatic storytelling applications that satisfies both users and developers, what is tested in the example application presented in chapter 4. This application, far for being a substitute from human creativity, acts as a promoter of the author efforts, applying the logical and combinatorial power of the computer to the generation of new and valuable stories based on the declarative and operational knowledge, always human-created, of the knowledge base.

On top of this application many automatic tests have been performed in order to obtain reliable data and to extract conclusions.

Conclusion obtained from the data is positive in the sense of it is possible to develop applications that efficiently generate satisfactory stories according to previously defined metrics. Limits will be those defined in the metrics and domain components'model of knowledge used in final applications.

Future work

With the intention of completing these conclusions, in this section some promising lines of research are presented, what can be useful for guiding future work in this field.

On one hand it is possible to continue tasks related with maintaining, fixing bugs and improving the software generated in this research project and its documentation. It is important to consider that the use of *DLApplication* for developing new automatic storytelling applications will allow the concepts of this thesis to become more mature, improving the credibility of this work's evaluation. New generic and specific domain components will also complete the repository of the framework, making more obvious its reusability and practical utility. Every tutorial, installer, manual or use case added to the project will represent benefits for the software and the ideas that support it, in the sense of spreading results and helping developers and researches in the field.

On the other hand, the exploration of questions about multimodal presentation of stories and interactive storytelling are the most promising research

lines in order to find possible industrial applications for this work. Connections with natural language generators can be improved adding hypertext, or predefined graphics and sounds to the generated stories. Even connections with virtual environments can be considered for representing drama using up-to-date 3D engines or interactive storytelling using advanced features of these environments as realistic physical simulation, autonomous or semiautonomous characters, real time, etc.

Future development of authoring tools for this framework, considered out of the scope of this work, will be decisive for its success in a more practical context. Development at the industrial level of automatic storytelling applications should delegate the creation of resources and initial knowledge base to experts in the specific domains of the application (scriptwriters, teachers, artists, etc.). There is no reason why these experts must know how to use sophisticated knowledge engineering or programming tools, being more reasonable to let them use authoring tools for their high level specific-domain tasks.

User modelling techniques are out of the scope of this research, but they are possible tools for adjusting result to the subjective criterion about value and novelty of each user. Constraint programming has been also suggested as a possible solution for the integration between generation and evaluation of abstract stories in the same problem-solving method for extending operational semantics of the components.

Although we are encouraged by the advances in questions of computational authorship, each time more evident, the fact that artificial intelligence and creativity are still under deep research does not means that automatic storytelling development should be unattended at the moment; indeed, even under the assumption that the perfect replication of human talent was not possible, the development of automatic storytelling application is justified. As Janet Murray remember us, patterns and formulas repetition existed since the very beginning of Art and Culture, performing a not worthless function that complete and make the most of more outstanding and revolutionary creations. This function is just the establishment of the particular conventions of a discipline into a specific historical context, setting implicit limits, each time more demanding, that the human inventiveness must overcome for being recognized as such.

Apéndice C

Abreviaciones

ABox Caja asertiva (en inglés *Assertional Box*)

API Interfaz de programación de aplicaciones (en inglés *Application Programming Interface*)

CB Base de casos (en inglés *Case Base*)

CBR Razonamiento basado en casos (en inglés *Case Base Reasoning*). Puede ir precedido del prefijo KI- denotando que dicho razonamiento hace uso intensivo de conocimiento (en inglés *Knowledge-Intensive*)

CWA Premisa de mundo cerrado (en inglés *Close World Assumption*)

DIG Grupo de implementación de lógica descriptiva (en inglés *Description Logic Implementation Group*). También hace referencia al lenguaje del mismo nombre desarrollado por dicho grupo

DL Lógica descriptiva (en inglés *Description Logic*)

IA Inteligencia Artificial

KB Base de conocimiento (en inglés *Knowledge Base*)

NLG Generación de lenguaje natural (en inglés *Natural Language Generation*)

NLP Procesamiento de lenguaje natural (en inglés *Natural Language Processing*)

NLU Comprensión de lenguaje natural (en inglés *Natural Language Understanding*)

OWA Premisa de mundo abierto (en inglés *Open World Assumption*)

OWL Lenguaje de ontologías para la web (en inglés *Web Ontology Language*, un acrónimo reordenado a propósito). Puede ir seguido de Lite, DL o Full, denotando un sublenguaje específico de OWL.

POO Programación orientada a objetos

PSM Método de resolución de un problema (en inglés *Problem-Solving Method*)

RBox Caja relacional (en inglés *Relational Box*)

RDF Marco de descripción de recursos (en inglés *Resource Description Framework*). Puede ir seguido del sufijo S o de Schema, denotando su metalenguaje de descripción

SW Web Semántica (en inglés *Semantic Web*)

SWRL Lenguaje de reglas para la Web Semántica (en inglés *Semantic Web Rule Language*)

TBox Caja terminológica (en inglés *Terminological Box*)

W3C Consorcio de la red mundial (en inglés *World Wide Web Consortium*)

Apéndice D

Glosario

Partiendo de expresiones comúnmente aceptadas, propuestas por autores como Szilas [Szi99] o Murray [Mur97], y haciendo uso del diccionario de la Real Academia Española de la Lengua [RAE], este glosario recoge los términos más relevantes para comprender este trabajo de tesis. En la medida de lo posible se usan términos españoles, evitando los extranjeros, salvo casos donde claramente el término extranjero se haya impuesto sobre los demás.

Autoría La función más importante del narrador, por la cual este establece el propósito de la historia, así como los criterios de valor y novedad que han de aplicarse sobre la misma.

Base de casos Un tipo particular de base de conocimiento que contiene casos para ser usados por un sistema CBR.

Base de conocimiento En el ámbito de este trabajo, significa un conjunto de asertos sobre individuos y sus relaciones en lógicas descriptivas. Los tipos de los individuos y los roles que juegan se encuentran siempre en una ontología asociada a la base de conocimiento.

Robot de charla Androide capaz de simular un diálogo con el usuario mediante el lenguaje natural.

Componente Un componente de software es un recurso independiente desarrollado para un fin concreto, que puede formar, junto a otros componentes, un sistema informático. Cada componente tiene su propia estructura e implementación, aunque cumple con una cierta interfaz que le permite ser conectado con otros componentes y reutilizado en diversas aplicaciones. Los componentes ontológicos presentados en este trabajo tiene la peculiaridad de contener, además de objetos con semántica operacional, ontologías con semántica declarativa.

Discurso La estructura que se le da a una parte de la fábula de una historia para poder narrarla después.

Drama Según Szilas, aquel estilo de narración (típicamente teatro y cine) donde la acción se representa de manera directa ante el espectador. Esto descarta la novela y el cuento como géneros propiamente dramáticos mientras que permite incluir en la definición todos aquellos juegos de mesa u ordenador que representen en estilo directo el desarrollo de una historia.

Fábula El contenido de una historia referente a los hechos ocurridos en el mundo ficticio de la misma.

Gramática de una historia Gerald Prince define la gramática de una historia (en inglés *story grammar*) como una serie de sentencias y fórmulas interrelacionadas mediante un conjunto de reglas para construir un conjunto de historias.

Historia La Real Academia Española define el término como: *"Aquella relación rigurosa de los hechos que conforman alguna aventura o suceso"* [RAE]. Este trabajo contempla otra definición de historia con un mayor énfasis en la importancia del argumento: una historia es el relato de una sucesión de hechos que hace énfasis en la causalidad.

Llamamos historia digital a aquella historia cuyo soporte principal es el medio informático.

Llamamos historia abstracta a la composición de una fábula y un discurso sin usar ningún medio de presentación concreto para la historia.

Inmersión En el sentido literal la inmersión es la *"acción de introducir o introducirse plenamente alguien en un ambiente determinado"* [RAE]. En este trabajo la inmersión no sólo hace referencia a la calidad estética del ambiente sino también a su valor narrativo y a la capacidad del interactor de participar como un agente más en dicho ambiente. Es uno de los placeres específicos de los entornos digitales que propone Murray. En un sentido más amplio este término ser refiere a la experiencia de asistir (de forma activa o pasiva) a unos sucesos que sabemos ficticios a pesar de que los sentidos nos dicen que son reales.

Literatura La literatura es el *"arte que emplea como medio de expresión una lengua"* [RAE]. Habitualmente el cine y el teatro no son considerados literatura, ya que su medio de expresión principal es la imagen y la representación dramática, respectivamente.

Narración Toda aquella expresión artística y social mediante la que se cuenta o refiere lo sucedido, ya sea esto real o ficticio. El término *storytelling*, muy utilizado, se traduce al castellano por "narrar o contar historias". El término Narración Digital (*Digital Storytelling*) es el que utilizan en el centro de investigación *ZGDV* para referirse a un tipo de narración interactiva.

Narrador Sujeto que realiza el acto narrativo. Normalmente ejerce las funciones activas en el acto de la narración. Estas funciones son tres: creación de la fábula, construcción de la historia y realización de la presentación. Se denomina *narrador implícito* al verdadero constructor del discurso de una historia, mientras que se denomina *narrador explícito* al que, en el contexto de dicho discurso, figura como sujeto del acto narrativo.

Novedad La novedad de una historia es el grado de diferencia que guarda con respecto a otros historias con las que es comparada. En el contexto del armazón propuesto, la novedad es un número real comprendido entre 0 y 1 que representa la diferencia entre una historia abstracta generada y sus distintas partes con respecto a todos los individuos de diversos dominios presentes en la base de conocimiento en el momento de su generación.

Ontología En el ámbito de este trabajo, significa un conjunto de conceptos, roles y axiomas en lógicas descriptivas. Distinguimos entre ontologías simples o compuestas, según el contenido de la ontología esté ubicado en un único espacio de nombres o en varios que hayan sido importados de otras ontologías. Las ontologías compuestas contienen algunas ontologías (también simples o compuestas) que pueden usarse de forma independiente.

Personaje Cada uno de los seres humanos, sobrenaturales, simbólicos, etc., que intervienen en la narración de una historia.

Presentación La parte de la historia referente al medio narrativo concreto utilizado para narrarla, como puede ser el texto, la imagen o el sonido.

Propósito El propósito de una historia representa la intención del autor expresada en términos de un dominio de conocimiento distinto al narrativo, y relacionada con la historia abstracta para crear un todo.

Público Grupo de personas a las que va destinada una historia. Normalmente ejercen la función pasiva en el acto de la narración, aunque en el

caso de la narración interactiva dicha función sea compatible con otra más activa y propia de los narradores, como es la de participar en la elaboración de la historia.

Valor El valor de una historia es el interés que tiene según el criterio establecido por el autor de la misma. En el contexto del armazón propuesto, el valor es un número real comprendido entre 0 y 1 que representa cómo encaja la historia abstracta generada y sus distintas partes con las definiciones preestablecidas sobre lo que es considerado "individuo valioso" en cada dominio.

Web Semántica Una extensión, en proceso de desarrollo, de la actual *World Wide Web* donde la información está complementada con anotaciones semánticas sobre los que distintos sistemas informáticos puede razonar, formando una gran base de conocimiento distribuida y en continuo crecimiento. Actualmente se conocen como tecnologías de la Web Semántica a las herramientas que utilizan los diversos estándares propuestos para esta iniciativa por el *Consorcio de la red mundial*.