UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE INFORMÁTICA

TESIS DOCTORAL

Un modelo de generación automática de historias con múltiples tramas

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Eugenio Pablo Concepción Cuevas

Directores

Pablo Gervás Gómez-Navarro
Gonzalo Rubén Méndez Pozo

Madrid

UNIVERSIDAD COMPLUTENSE DE MADRID
FACULTAD DE INFORMÁTICA

TESIS DOCTORAL

Un modelo de generación automática de historias con múltiples tramas

MEMORIA PARA OPTAR AL GRADO DE DOCTOR

PRESENTADA POR

Eugenio Pablo Concepción Cuevas

DIRECTORES

Pablo Gervás Gómez-Navarro
Gonzalo Rubén Méndez Pozo

Un modelo de generación automática de historias con múltiples tramas

TESIS DOCTORAL

Autor
Eugenio Pablo Concepción Cuevas

Directores
Pablo Gervás Gómez-Navarro
Gonzalo Rubén Méndez Pozo

Doctorado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

Un modelo de generación automática de historias con múltiples tramas

Tesis Doctoral en Ingeniería Informática

Autor
Eugenio Pablo Concepción Cuevas

Directores
Pablo Gervás Gómez-Navarro
Gonzalo Rubén Méndez Pozo

Doctorado en Ingeniería Informática
Facultad de Informática
Universidad Complutense de Madrid

8 de mayo de 2023

UNIVERSIDAD COMPLUTENSE MADRID

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD DE LA TESIS PRESENTADA PARA OBTENER EL TÍTULO DE DOCTOR

D./Dña. EUGENIO PABLO CONCEPCIÓN CUEVAS,
estudiante en el Programa de Doctorado EN INGENIERÍA INFORMÁTICA (RD 99/2011),
de la Facultad de Informática de la Universidad Complutense de Madrid, como autor/a de la tesis presentada para la obtención del título de Doctor y titulada:

UN MODELO DE GENERACIÓN AUTOMÁTICA DE HISTORIAS CON MÚLTIPLES TRAMAS

y dirigida por: PABLO GERVÁS GÓMEZ-NAVARRO Y GONZALO RUBÉN MÉNDEZ POZO

DECLARO QUE:

La tesis es una obra original que no infringe los derechos de propiedad intelectual ni los derechos de propiedad industrial u otros, de acuerdo con el ordenamiento jurídico vigente, en particular, la Ley de Propiedad Intelectual (R.D. legislativo 1/1996, de 12 de abril, por el que se aprueba el texto refundido de la Ley de Propiedad Intelectual, modificado por la Ley 2/2019, de 1 de marzo, regularizando, aclarando y armonizando las disposiciones legales vigentes sobre la materia), en particular, las disposiciones referidas al derecho de cita.

Del mismo modo, asumo frente a la Universidad cualquier responsabilidad que pudiera derivarse de la autoría o falta de originalidad del contenido de la tesis presentada de conformidad con el ordenamiento jurídico vigente.

En Madrid, a 5 de diciembre de 2022

Fdo.: _____

Esta DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD debe ser insertada en la primera página de la tesis presentada para la obtención del título de Doctor.

Dedicatoria

A mis hijas, Lidia y Victoria, Victoria y Lidia, alegría de mis días.

Agradecimientos

Escribir una tesis doctoral es, pese a lo que pueda parecer a priori, un esfuerzo colectivo. Es el esfuerzo del doctorando, que lleva a cabo la investigación; pero también lo es de sus directores, que tratan de encauzarlo; y de sus seres queridos, que tratan de que no ceda al desánimo; y de sus compañeros de trabajo, que le animan y se interesan por lo que hace; y de sus compañeros de investigación, que le guían y aconsejan; y así en una cadena que involucra a tantas y tantas personas.

Mi agradecimiento se extiende por tanto a muchas personas que han formado o forman parte de mi vida, y que me han ayudado de múltiples formas en estos largos años de estudio.

Quisiera empezar dando las gracias a mis directores, Pablo y Gonzalo, a quienes considero ante todo amigos. Siempre he encontrado orientación en sus consejos y comentarios, y siempre he encontrado comprensión y solidaridad en sus respuestas. A pesar de las dificultades de dirigir a un doctorando con una vida laboral y personal tan complicadas, siempre han mostrado el mejor ánimo cuando les he necesitado.

Quisiera agradecer a mis hijas Lidia y Victoria su cariño, el gesto sencillo de estar a mi lado e iluminar mis días con sus sonrisas. No tengo claro qué pensarán de ver a su padre metido en semejantes entuertos, pero no tengo dudas de que ellas son un faro en mi vida. También quisiera agradecer a Ruth, su madre, por haber escrito tantas páginas felices de mi vida y haberme apoyado durante todo el tiempo que estuvimos juntos.

Quisiera agradecer también a mi madre, mi padre, mis hermanas y mi hermano, mis tías y mi prima Rosa su cariño y apoyo durante todo este tiempo. Y en particular, a mi hermana Elena, decana de la Academia en la familia, sus consejos de veterana.

Finalmente, quisiera dar las gracias a muchos y muy buenos amigos y amigas, que me han ayudado y apoyado: a Carlos, por tantos y tan buenos consejos a lo largo de la investigación; a Iñaki, por haberme animado y dado tan buenos consejos sobre Neo4J; a Juanqui, por su ánimo en los años más duros; a Celia, que desde Dallas me ha mandado siempre buenas vibraciones; a Yoli, que tan buenos momentos compartimos en el trabajo; a Marcos, por sus chistes malos y geniales; a Sara, que siempre te ayuda con una sonrisa; a Almudena, por su sosegada compañía; a Raúl, que siempre me da ánimos; a Ana, porque siempre se acuerda de mí; a Rocío, por mandarme la energía de un saiyan; a mis alumnos y amigos Markel y Juan, que

son dos fueras de serie; a Óscar, que siempre me animó a rematar la jugada; a todos los compañeros y compañeras restantes del Grupo NIL, que además de haber proporcionado consejo, también han demostrado ser buenos amigos; y finalmente, una mención especial a todos los Académicos Anónimos, a quienes debo una cerveza (o dos).

Y por último, gracias a ti, que estás leyendo este trabajo, por tu tiempo y tu interés.

Resumen

Desde los primeros trabajos en Inteligencia Artificial, ha habido un esfuerzo incesante por replicar de la mejor manera posible los mecanismos de la inteligencia y la creatividad humanas. La generación automática de historias es un campo de la Creatividad Computacional, un área de investigación dentro de la Inteligencia Artificial que trata de recrear estos mecanismos de la creatividad humana en sistemas computacionales. En el caso de la generación automática de historias, el objetivo que se persigue es la producción de artefactos que puedan asemejarse a la literatura creada por seres humanos.

En este caso en particular, el esfuerzo se ha centrado en aquellas historias que albergan más de una trama. Un ejemplo de este tipo de historias lo tenemos en las novelas-río, término procedente del francés *roman-fleuve*, que designa novelas de larga duración, habitualmente agrupadas en una colección, que relatan historias de varios personajes que confluyen hacia un punto de encuentro, manteniendo una consistencia global. Otro ejemplo de historias con múltiples tramas lo tenemos en obras clásicas como *Don Quijote de la Mancha*, o *Las mil y una noches*, donde se anidan historias dentro de historias, como una secuencia de matrioshkas.

El presente trabajo resume el esfuerzo investigador realizado generación automática de historias para permitir la creación de sistemas generadores capaces de producir historias con varias tramas. Entendiendo que estas historias deben construirse en términos de un modelo de representación del conocimiento acorde, y mediante un procedimiento específico que permita combinar varias tramas en una única historia; el resultado es el trabajo presentado en esta tesis. La investigación se ha centrado en tres aspectos clave: la representación del conocimiento en el dominio de las historias, la arquitectura de sistemas generadores y la generación de historias multitrama.

Palabras clave

Generación automática de historias, Creatividad Computacional, Inteligencia Artificial, Representación del Conocimiento, Arquitectura del Software, Generación de historias multitrama.

Abstract

Since the earliest work in Artificial Intelligence, there has been a relentless effort to replicate as best as possible the mechanisms of human intelligence and creativity. Automatic story generation is a field of Computational Creativity, a research area within Artificial Intelligence that tries to recreate these mechanisms of human creativity in computational systems. In the case of automatic story generation, the goal is to produce artefacts that can resemble literature created by human beings.

In this particular case, the focus has been on stories that contain more than one plot. An example of this type of story is the river-novel, a term from the French *roman-fleuve*, which designates novels of long duration, usually grouped in a collection, which tell the stories of several characters that converge towards a meeting point, maintaining an overall consistency. Another example of stories with multiple plots can be found in classic works such as Don Quixote, or The Arabian Nights, where stories are nested within stories, like a sequence of matrioshkas.

The present work summarises the research effort made in automatic story generation to enable the creation of generative systems capable of producing stories with several plots. Understanding that these stories must be constructed in terms of a matching knowledge representation model, and by means of a specific procedure that allows to combine several plots into a single story, the result is the work presented in this thesis. The research has focused on three key aspects: the representation of knowledge in the story domain, the architecture of generator systems and the generation of multi-frame stories.

Keywords

Automatic Story Generation, Computational Creativity, Artificial Intelligence, Knowledge Representation, Software Architecture, Muti-plot story generation.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

"El programa que tiene en la cabeza un poeta corriente está creado por la civilización en cuyo medio ha nacido, la cual, a su vez, ha sido preparada por la que la precedió; esta última, por otra, más temprana todavía, y así, hasta los mismos comienzos del Universo [...]. Para programar la máquina hacía falta, pues, volver a repetir antes, si no todo el Cosmos desde el principio, por lo menos una buena parte de él"
— Stanislaw Lem

1.1. Contexto

La creación y narración de historias se perciben actualmente como un componente fundamental de la capacidad cognitiva humana (Thorndyke, 1977a) y como una herramienta crucial para participar con éxito en la sociedad moderna (Marsella et al., 2004). También fue uno de los primeros objetivos que se fijaron en el campo de la Inteligencia Artificial, cuando los investigadores comenzaron a intentar modelar las capacidades humanas de forma computacional, y sigue siendo un campo de investigación activo en la actualidad (Gervás et al., 2019).

De acuerdo con la definición de Gervás (2012):

> Un algoritmo generador de historias se refiere a un procedimiento computacional que da lugar a un artefacto que puede considerarse una historia.

En este sentido, el término sistema de generación de historias (*Story Generation System*) puede considerarse prácticamente como un sinónimo de sistema narrativo (*Storytelling System*), es decir, un sistema computacional diseñado para contar historias.

1.2. Motivación

Imaginemos de forma hipotética que quisiéramos desarrollar un sistema para generar historias que fuera capaz de generar una historia como "Guerra y paz" (Tolstoy, 2017). Esta novela cuenta la historia de la invasión napoleónica de Rusia y su impacto en la sociedad rusa a través de las historias de cinco familias aristocráticas rusas. La trama de la historia es un complejo tejido de líneas argumentales o subtramas que se entrelazan y nos permiten seguir a los personajes en su interacción con el mundo hasta completar una convergencia de las mismas.

Teniendo en cuenta lo anterior, crear un sistema de este tipo sería un reto formidable. Semejante generador de historias debería ser capaz de operar a diferentes niveles de detalle y resolver muchas subtramas entrelazadas. Para mantener la consistencia global de la historia sería necesario que el encaje entre los acontecimientos de cada capítulo en una subtrama que afectase a otra fueran coherentes. Si, por ejemplo, un personaje parte de una localización a otra en una subtrama, no tendría sentido que volviera a aparecer en el mismo lugar en una escena de otra subtrama que tiene lugar en el tiempo inmediatamente después de la partida. Por otro lado, si las subtramas no guardan elementos en común entre sí, el potencial lector no sería capaz de entender que ambas subtramas pertenecen a la misma historia.

Los ejemplos anteriores no agotan la lista de requisitos; esta novela es una obra verdaderamente monumental, contiene varios capítulos dedicados a diversos aspectos del conocimiento humano. El conocimiento relativo al mundo de la historia que necesitaría tener el sistema generador sería otro reto. En una novela como "Guerra y paz" hay múltiples aspectos que requerirían un tratamiento específico en un sistema de generación de historias. Probablemente, esto requeriría el desarrollo de una arquitectura del sistema que fuera extensible para poder ir agregando componentes especializados que trabajaran en cada aspecto. Y disponer de un modelo de representación del conocimiento que permitiera un intercambio efectivo entre dichos componentes.

Las ideas expuestas en esta introducción son las que han guiado el trabajo desarrollado a lo largo de la presente tesis. Si bien la realidad actual de la generación automática de historias dista bastante de llegar a semejantes cumbres, la posibilidad de combinar varias tramas, integrar componentes diversos, y disponer de mecanismos extensibles de representación del conocimiento, abren la puerta a la creación de historias más ricas y complejas, dando un paso para aumentar el interés y la variedad de los resultados.

1.3. Objetivos específicos

Esta sección recoge los objetivos específicos de la presente tesis. Estos se han agrupado en dos categorías principales: objetivos científicos, que definen la orientación del trabajo investigador realizado; y objetivos tecnológicos, que son la expresión técnica de dicho trabajo.

1.3.1. Objetivos científicos

Se han perseguido los siguientes objetivos científicos a lo largo del proceso de investigación:

- Objetivo C1: Desarrollar un modelo de representación del conocimiento que permita representar historias con múltiples tramas
- Objetivo C2: Desarrollar una arquitectura de referencia que proporcione flexibilidad para la creación de sistemas de generación de historias con múltiples tramas
- Objetivo C3: Diseñar mecanismos para generar automáticamente historias con múltiples tramas

1.3.1.1. Objetivo C1: Desarrollar un modelo de representación del conocimiento que permita representar historias con múltiples tramas

Este objetivo persigue el desarrollo de un modelo de representación de historias, desde la perspectiva de la generación automática, que contengan varias tramas. Este objetivo general puede segregarse en la consecución de varios subobjetivos. Primero, la definición de un modelo que contenga todos los aspectos y dimensiones relevantes relativos al concepto de historia en generación automática. Para lograr dicho objetivo, se considera recomendable realizar previamente un análisis de los modelos existentes en el ecosistema de sistemas generadores, de forma que se puedan extraer las características comunes relevantes. En segundo lugar, habiendo definido un modelo de representación suficientemente completo y expresivo, se debería perseguir que dicho modelo sea capaz de representar adecuadamente el concepto de historia con múltiples tramas. El concepto de trama se refiere a la secuenciación de los hechos que tienen lugar en una historia y que definen la estructura de su contenido. Además de los objetivos expresados, hay otro condicionante implícito, relativo a la compaginación del objetivo C1 con el resto, la adecuación del modelo de representación a la operación de los resultados perseguidos en los otros objetivos.

1.3.1.2. Objetivo C2: Desarrollar una arquitectura de referencia que proporcione flexibilidad para la creación de sistemas de generación de historias con múltiples tramas

El resultado perseguido por este objetivo consiste en disponer de un modelo arquitectónico flexible que soporte la creación de sistemas generadores. Esto quiere decir que el resultado que se busca no es un sistema generador específico, sino un marco arquitectónico de referencia que sea aplicable al diseño de la arquitectura de un sistema generador de historias. Por tanto, la solución propuesta debe ofrecer un planteamiento abstracto que pueda trasladarse a una implementación, y que los sistemas generadores resultantes sean capaces de generar historias con múltiples tramas. Nuevamente, este objetivo conlleva otros subobjetivos, como el análisis de los sistemas generadores existentes para comprender su operación y extraer los *drivers*

arquitectónicos que guiarán el diseño de la arquitectura de referencia. Además, la solución planteada deberá ser compatible con los otros objetivos, particularmente con C1.

1.3.1.3. Objetivo C3: Diseñar mecanismos para generar automáticamente historias con múltiples tramas

Este objetivo persigue el diseño de técnicas que permitan generar historias con múltiples tramas. Al igual que en el caso de los dos objetivos anteriores, conlleva la consecución de una serie de objetivos. En primer lugar, implica la comprensión de los mecanismos literarios asociados a la creación de narraciones con múltiples tramas. A partir de dicha comprensión, el siguiente subobjetivo se centra en la definición de heurísticas que puedan ejecutarse por medios computacionales y que permitan trabajar con múltiples tramas en una historia. Una vez definidas las heurísticas, será necesario desarrollar los sistemas o componentes que las implementen. Y finalmente, recopilar los resultados obtenidos y evaluarlos de forma cualitativa para determinar si la implementación y el diseño de las heurísticas conducen a la creación automática de historias no triviales con múltiples tramas.

1.3.2. Objetivos tecnológicos

Como soporte para alcanzar los objetivos científicos, los trabajos de diseño y construcción han perseguido los siguientes objetivos tecnológicos:

- Objetivo T1: Desarrollar sistemas que permitan poner a prueba la arquitectura diseñada
- Ojetivo T2: Desarrollar sistemas que implementen los mecanismos de generación de historias con múltiples tramas

1.3.2.1. Objetivo T1: Desarrollar sistemas que permitan poner a prueba la arquitectura diseñada

Este objetivo tecnológico sirve realmente como respaldo del objetivo científico C2 e indirectamente al C1. En este caso, la consecución del objetivo implica el desarrollo de varios sistemas generadores de historias que se adhieran a la arquitectura de referencia obtenida como resultado de la consecución del objetivo C2. El propósito de dichas implementaciones es verificar que el diseño de la arquitectura de referencia se ha ajustado a los *drivers* arquitectónicos establecidos. Y también, verificar que los atributos de cualidad del sistema derivados de la aplicación de los *drivers* al diseño se verifican en la implementación de la arquitectura de referencia obtenida. Queda incluida en dicha validación que el modelo de representación del conocimiento (Objetivo C1) es adecuado y compatible con el diseño arquitectónico.

1.3.2.2. Objetivo T2: Desarrollar sistemas que implementen los mecanismos de generación de historias con múltiples tramas

Este objetivo de referencia se deriva principalmente del objetivo científico C3. Lo que persigue el objetivo T2 es asegurar que las heurísticas obtenidas como resultado de la consecución de C3 son computacionalmente viables. De poco serviría definir mecanismos de entretejido de múltiples tramas si su implementación tuviese un coste de complejidad que hiciera inviable la ejecución de los mismos en un entorno real. El objetivo T2 se centra en implementar las heurísticas y verificar que se pueden obtener resultados en el marco de la arquitectura de referencia y el modelo de representación del conocimiento desarrollados por la tesis.

1.4. Relevancia de la investigación

La relevancia de la presente investigación se puede articular en tres aspectos, derivados de los objetivos científicos expresados: representación del conocimiento, arquitectura de referencia y mecanismos de generación de múltiples tramas.

1.4.1. Representación del conocimiento

El problema de cuánto conocimiento es necesario para generar una historia que se asemeje a la creada por un ser humano es un tema clásico en el área de la Generación Automática de historias. Desde los años 70 se viene buscando una solución a esta cuestión. El propósito de la representación del conocimiento y el razonamiento sobre el mismo en computación es comprender tan bien la naturaleza de la inteligencia y la cognición humanas que se puede lograr que los sistemas computacionales muestren capacidades similares a las mismas (Van Harmelen et al., 2008).

Sumado al problema del conocimiento mínimo necesario para generar una historia de calidad está el de la representación del propio conocimiento. Y, como bien anticipaba Stanislaw Lem en su relato "*El electrobardo de Trurl*", perteneciente a su genial obra "*Ciberíada*" (Lem, 2021), la creación literaria, llevada hasta sus últimos extremos, requeriría de la recreación del universo en su totalidad.

Los sistemas de generación de historias se enfrentan al importante reto de adquirir recursos de conocimiento en los formatos de representación particulares que utilizan. Se enfrentan a una dificultad inherente al uso de lenguajes formales en la separación entre la formulación de las necesidades en el mundo real y su representación en una construcción formal. La representación es, por tanto, otra dificultad en cuanto a la forma de aplicar el conocimiento a la generación.

Los esfuerzos investigadores en generación automática de historias acumulan ya más de 50 años de resultados. Si bien los sistemas generadores han experimentado una continua mejora a lo largo de la evolución de la disciplina, hay una constante que se ha mantenido a lo largo de todo este tiempo: la dependencia del conocimiento para generar buenas historias, entendiendo buenas historias por historias que se asemejen lo más posible a la creación humana. Trasladando este concepto a la creación artística por un autor humano, la experiencia vital del mismo es clave en la calidad y la

orientación de su obra. No existe una respuesta única al problema de cuál es el conocimiento mínimo necesario para crear historias, al igual que tampoco existe un sistema que emule completamente todas las capacidades humanas en la creación literaria. Cada sistema generador, o familia de sistemas generadores, ha concentrado sus esfuerzos en definir ese conjunto mínimo de datos necesarios. El enfoque que se ha buscado en el presente trabajo ha sido tratar de proporcionar un mecanismo que maximice la compatibilidad entre los diferentes sistemas generadores y sus modelos de conocimiento, de forma que se pueda asegurar su interoperabilidad.

En general, los sistemas de generación de historias dependen en gran medida del conocimiento para llevar a cabo sus objetivos. Utilizando las palabras de Natalie Dehn, tal y como expresó el problema de la representación del conocimiento en su tesis (Dehn, 1981a),

> Un paso importante en la construcción real de la memoria inicial de un generador de historias es determinar qué conocimiento previo debe ir en ella.

La representación del conocimiento propuesta en este trabajo para las historias y sus conceptos relacionados (trama, personajes, discurso, etc.) está fuertemente influenciada por la Narratología, una disciplina de las humanidades dedicada al estudio de la lógica, los principios y las prácticas de la representación narrativa (Margolin et al., 2013). Muchos de los conceptos han sido definidos a partir de elementos de la Narratología clásica (Barthes, 1980; Chatman, 1980; Genette, 1980) y se han adaptado para cumplir con los objetivos específicos enunciados anteriormente.

1.4.2. Arquitectura para la generación de historias

A pesar de que la primera mención del término "arquitectura del software" data del año 1969, cuando Ian P. Sharp lo empleó en una conferencia auspiciada por la OTAN sobre técnicas aplicadas de Ingeniería del Software, (Randell y Buxton, 1970); realmente la formalización del área como disciplina es mucho más reciente y se debe principalmente a los trabajos de Shaw, Garlan y Clements (Shaw y Garlan, 1996; Bass et al., 2003; Clements et al., 2003; Shaw y Clements, 2006).

Se define la arquitectura del software de un sistema como "*la estructura del sistema, que comprende elementos software, las propiedades de esos elementos visibles externamente y las relaciones entre ellos*", (Bass et al., 2003). Expresado de otra forma, la arquitectura software se puede ver como la estructura que soporta y dota de orden interno a cualquier sistema software, tal como hace su equivalente en la edificación. Esta definición implica que la arquitectura se ocupa de la definición de los componentes y de la forma en que estos se ensamblan e interactúan, pero sin entrar a describir los detalles internos de los mismos.

Un estilo arquitectónico "*Establece un conjunto predeterminado de tipos de elementos, especifica sus responsabilidades e incluye reglas y guías para organizar las relaciones entre estos*" (Rozanski y Woods, 2011).

El estilo de diseño adoptado para el planteamiento de la arquitectura de referencia y sus diversas implementaciones ha sido el de una arquitectura basada en

microservicios (Newman, 2015; Wolff, 2016; Concepción et al., 2017c).

Las razones para haber adoptado un diseño basado en microservicios se debe a que estos proporcionan varios atributos de calidad perseguidos en los objetivos de la presente investigación. Estas características son especialmente deseables para el propósito deseado con la arquitectura, pues permiten realizar cambios y extensiones de los diversos componentes con vistas a refinar el proceso de generación, y especialmente con el desarrollo del proceso de generación de historias con múltiples tramas, como se puede ver en los artículos correspondientes al Capítulo 4.

1.4.3. Generación de historias con múltiples tramas

Muchas de las formas narrativas que destacan por su valor de entretenimiento en la cultura occidental (comedias de Shakespeare, películas de Hollywood, series de televisión, novelas del siglo XIX) comparten la característica de incluir varias líneas argumentales que se combinan para darles interés. Muchas de estas formas establecidas se basan en mecanismos para combinar ingeniosamente varios hilos narrativos en una única secuencia lineal, de modo que el público pueda seguir fácilmente la historia general.

Existen mecanismos muy claros para articular este planteamiento en un discurso narrativo interesante y atractivo para el consumidor.

Sin embargo, habitualmente en generación automática los esfuerzos por modelar computacionalmente la generación de historias se han centrado principalmente en la construcción de historias con una sola línea argumental. Esto se debe en parte a la aplicación de una regla tradicional de la ingeniería: no considerar versiones complejas del problema hasta que se hayan resuelto las versiones sencillas; y, en parte, a la aplicación de otra regla: si el artefacto que se intenta construir se compone de varias partes, el proceso de construcción debería entender cuáles son esas partes elementales y cómo se unen. El presente trabajo explora soluciones computacionales a la tarea de construir una historia que combina más de una línea argumental en un único discurso lineal.

La combinación de varias líneas argumentales en un discurso es más compleja que un simple entretejido de las líneas. Las líneas argumentales no son en realidad flujos independientes de discurso narrativo que se combinan en un tejido complejo: los conjuntos de personajes de diferentes subtramas combinadas en una historia no permanecen separados, sino que pueden fusionarse, de modo que el mismo personaje de la historia general suele desempeñar papeles diferentes en más de una de las subtramas.

El proceso que queremos modelar implica una operación adicional de fusionar (algunos de los personajes) los hilos narrativos para las distintas líneas argumentales, instanciando sus roles narrativos (principales o secundarios) con el conjunto de personajes de la trama general.

1.5. Estructura del documento

La presente tesis, realizada en formato de compilación de artículos de investigación, se estructura de acuerdo a los siguientes contenidos:

- El Capítulo 1 se dedica a introducir el problema de investigación, el contexto científico-técnico al que pertenece, y una justificación de la aportación realizada por la investigación.
- El Capítulo 2 proporciona una descripción detallada del marco teórico de los sistemas de generación automática de historias, una historia de los hitos más relevantes para la investigación en este campo, y un análisis de los modelos de representación del conocimiento. Todos estos elementos tienen una relación con los aspectos tratados en la solución propuesta en los siguientes capítulo.
- El Capítulo 3 contiene los artículos relativos a la definición y el desarrollo del modelo de representación del conocimiento.
- El Capítulo 4 contiene los artículos relativos a la definición de la arquitectura de referencia.
- El Capítulo 5 contiene los artículos relativos al diseño e implementación de un sistema generador de historias con múltiples tramas basado en el modelo de representación y la arquitectura de referencia.
- El Capítulos 6 contiene una discusión relativa a los objetivos establecidos y a su correspondencia con las aportaciones de la investigación.
- El Capítulo 7 recoge las conclusiones y las posibles vías para un trabajo futuro.

Capítulo 2

Estado de la cuestión

"Ars longa, vita brevis"
— Hipócrates

2.1. Introducción

A principios del siglo pasado, Vladimir Propp, un folclorista ruso, desveló la estructura interna de los cuentos clásicos rusos tras haber identificado un conjunto de patrones recurrentes y haberlos formalizado en una representación abstracta (Propp, 1968). A pesar de no ser el primer intento de analizar la estructura interna de la narrativa, el esfuerzo de Propp tiene el mérito de ser la primera formalización de la estructura de los cuentos de una manera que se asemeja mucho a una abstracción computacional. Y, aunque Propp nunca describió su formalismo como una gramática para la generación de historias, autores como Turner (1993) lo mencionan como clara inspiración para su trabajo en generación automática. La búsqueda y aplicación de estrategias que permitieran formalizar los procesos internos de la literatura han sido constantes desde los pioneros en este campo, redescubiertos para la comunidad por Ryan (2017), hasta los más recientes trabajos.

Desde un punto de vista pragmático, la tarea de emular la creación literaria humana constituye un reto de ingeniería de primer orden. Mientras que los sistemas exitosos centrados en el proceso de generación se han logrado aplicando una o varias estrategias como, por ejemplo, la planificación (Ghallab et al., 2004), el razonamiento basado en casos razonamiento basado en casos (Kolodner, 2014), o la simulación basada en agentes (Chang y Soo, 2008), por citar algunos ejemplos; no hay demasiados casos de soluciones basadas en varios sistemas cooperando, siendo Slant (Montfort et al., 2013) un buen ejemplo de esto último. Es poco probable que una capacidad humana tan compleja pueda ser modelada en su totalidad mediante el recurso a un solo sistema. El que determinadas soluciones conduzcan a resultados valiosos, capturando algunas de las características que componen una historia válida, sugiere claramente que dichos sistemas tienen algo que contribuir a la tarea. Sin embargo, las soluciones en cada caso también muestran limitaciones en lo que respecta a las características de las historias que son capaces de generar, y dichas limitaciones vienen dadas por la

complejidad de la tarea a abordar y la necesidad de enfocar la solución en una parte de dicha complejidad (Concepción et al., 2017a, 2018a).

Esta sección se centra en la revisión de trabajos relacionados que son relevantes para el propósito de esta investigación. Comienza presentando algunos antecedentes relevantes en la narración automática y luego procede a revisar los esfuerzos de investigación que se han centrado específicamente en la generación de tramas múltiples.

2.2. Historia de los sistemas generadores de historias

La presente sección revisa de forma cronológica la historia de los sistemas generadores. La selección de los sistemas referidos se ha realizado con vistas a destacar las principales aportaciones de cada uno de los sistemas al conjunto de la generación automática de historias.

Con el fin de estructurar los sistemas destacados en esta cronología, se han agrupado en tres categorías: los **pioneros**, que recogen sistemas que habían caído en el olvido y que fueron intentos tempranos de generación automática; los **sistemas clásicos**, que están bien documentados, sentaron las bases de la operación y los aspectos esenciales de un sistema generador, y han influido en un buen número de sistemas posteriores; y finalmente, los **sistemas contemporáneos**, que combinan estrategias de los sistemas clásicos y desarrollan nuevos enfoques para generar historias más ricas y complejas.

2.2.1. Los pioneros

Gracias al trabajo de Ryan (2017), se ha podido redescubrir la historia de los pioneros en la generación automática de historias. A pesar de lo que se ha pensado durante mucho tiempo, los primeros sistemas generadores de historias se remontan a los años 60, una década antes de Novel Writer (Klein, 1973).

En 1960, el MIT publicó un memorando técnico sobre la arquitectura de un sistema llamado SAGA II (Morse, 1960). Este programa generaba guiones para programas de televisión occidentales mediante un enfoque probabilístico. El procedimiento de generación actuaba dividiendo cada escena en tiempos narrativos que se estructuraban como un grafo en forma de árbol definido por las ramas de acción. Esta estructura se recorría según una heurística probabilística basada en el estado de la historia.

En los primeros años de la década de 1960, Joseph E. Grimes, un lingüista que estudiaba las culturas nativas de México, desarrolló un sistema que aplicaba una simulación de Montecarlo para crear tramas de cuentos populares (Ryan, 2017). Su enfoque se basaba en el trabajo de Vladimir Propp sobre la estructura de los cuentos populares rusos (Propp, 1968). El procedimiento de generación tomaba como entrada un patrón de cuento básico y luego aplicaba una simulación de Montecarlo para generar historias con un componente aleatorio. La aleatoriedad se utilizaba para enlazar un episodio con otro, al igual que servía para seleccionar los papeles que desempeñaba cada personaje y la acción que determinaba si el episodio se cumplía.

Robert I. Binnick desarrolló en 1968 otro sistema de generación de historias

(Binnick, 1969) como proyecto paralelo al trabajo de Victor Yngve en generación de lenguaje natural (Yngve, 1961). Este sistema también se basaba en las funciones de los cuentos populares de Propp. El desarrollo era un "programa COMIT que instanciaba una gramática de estructura de frases (probablemente) libre de contexto" (Ryan, 2017), que podía generar esquemas argumentales.

2.2.2. Los generadores clásicos (1973-1998)

En esta categoría se han considerado aquellos sistemas generadores que desarrollaron los enfoques fundamentales en el campo de la generación automática de historias, tanto a nivel de estrategias de generación, como a nivel de modelos de representación del conocimiento. La organización de los sistemas considerados es cronológica, partiendo de los primeros sistemas generadores y continuando con aquellos sistemas que podrían considerarse precursores en sus respectivas estrategias y enfoques.

2.2.2.1. Novel Writer

Novel Writer (Klein, 1973) es el primer sistema de generación automática de historias formalmente documentado como tal. Novel Writer generaba historias de asesinatos en el contexto de una fiesta de fin de semana. Tomaba como entradas una descripción del mundo en el que sucedía la acción, junto con una completa caracterización de los personajes participantes (rasgos, relaciones y la predisposición a la violencia y al sexo). El proceso de generación consistía en dos algoritmos diferentes: un motor basado en reglas para implementar las transiciones de estado en el mundo ficticio, y una secuencia de escenas asociada al tipo de historia a desarrollar. Estos elementos eran bastante restrictivos, por lo que las historias generadas tenían una estructura idéntica y la principal diferencia provenía de las variaciones entre los personajes que desempeñaban los roles previstos en la trama.

2.2.2.2. Tale-Spin

Tale-Spin (Meehan, 1977) era un sistema generador de historias que inicialmente producía relatos sobre un conjunto de personajes que vivían en un bosque, pero que posteriormente demostró ser capaz de generar varias historias, con diferentes tonos y temáticas, sólo con cambiar el trasfondo (Wardrip-Fruin, 2009). Desde un punto de vista técnico, Tale-Spin es un planificador, es decir, un sistema de resolución de problemas, que seguía un enfoque descendente para cumplir los objetivos definidos. Los objetivos de los personajes eran las entradas que el sistema utilizaba para explorar el espacio de soluciones y encontrar los posibles subobjetivos. La elección de las acciones que los personajes podían realizar determinaba cómo alcanzar cada objetivo individual. El sistema tomaba después la secuencia resultante de las acciones de los personajes y las describía para construir la historia de salida.

Tale-Spin se diseñó originalmente para soportar tres modos de funcionamiento: dos modos interactivos y un modo autónomo. El funcionamiento interactivo requería que el usuario humano definiera ciertas características del mundo de la historia para

que el sistema pudiera generarla. Por el contrario, el modo autónomo permitía al programa configurar directamente el mundo para generar un determinado tipo de historias. Tale-Spin requería que el usuario introdujera los personajes que debía incluir en la historia. Tras el paso de selección de personajes, el sistema generaba un perfil del personaje, que incluía un conjunto básico de datos sobre él. Esta caracterización incluía información sobre los rasgos físicos del personaje y el entorno en el que vivía. Además, cada vez que el programa añadía un nuevo personaje, también actualizaba al resto de los personajes del reparto con el conocimiento de la existencia y ubicación de este nuevo participante.

2.2.2.3. Author

Author (Dehn, 1981a) fue el primer sistema generador que incluyó los objetivos del autor como parte del proceso de generación de historias. Dehn consideraba que las historias no son más que una justificación a posteriori de una trama concebida en la mente del autor. A partir de este planteamiento, Author fue desarrollado con el propósito de emular la mente de un escritor. Incluso cuando el autor no ha expresado claramente los objetivos o acontecimientos de la historia, hay objetivos subyacentes que impulsan el proceso de narración. En este nivel existen ciertas restricciones como la coherencia de la historia y la credibilidad de los personajes. Estas limitaciones implican la derivación de objetivos específicos para su cumplimiento. Por ello, Dehn explicaba que una historia se entiende como "la consecución de una compleja red de objetivos del autor" (Dehn, 1981a). Aunque estos objetivos son útiles durante el proceso de desarrollo, ya que ayudan a estructurar la historia, no son necesariamente evidentes en el producto final. Dehn daba especial importancia a las estrategias de búsqueda en la memoria y al recuerdo. La razón de ser de estos aspectos es la consideración de la autora de que el recuerdo desempeña dos funciones relevantes en el proceso de generación de historias: en primer lugar, como fuente de material externo relevante; y en segundo lugar, como forma de seguir la pista del material interno, los objetivos narrativos y otros aspectos.

Author era conceptualmente un planificador pero, a diferencia de Tale-Spin, utilizaba la planificación para cumplir los objetivos de autor en lugar de los objetivos del personaje. Durante la generación de la historia, Author procedía a través de un proceso iterativo que trabajaba continuamente en la mejora de la trama para cumplir con los objetivos del autor, que también podían ser reajustados en la iteración después de reflexionar sobre la historia en curso. Este mecanismo permitía rememorar periódicamente al autor los objetivos anteriores y decidir si se habían vuelto más relevantes de acuerdo con la evolución de la trama.

2.2.2.4. Roald

Inspirado en Tale-Spin, Masoud Yazdani desarrolló **Roald** (Yazdani, 1983), un sistema de narración para generar eventos para historias. Simulaba un mundo en el que los personajes reconocían y resolvían problemas dentro de las restricciones impuestas por el escritor.

Roald simulaba un mundo en el que los hechos que acontecían eran consecuencia del comportamiento de los personajes dirigido a un objetivo. La simulación se realiza en varios pasos. El primer paso era la producción de la representación del mundo. El usuario creaba un mundo y se proporcionaba un conocimiento parcial del mismo a cada uno de los personajes de la historia. El proceso de creación del mundo no tenía que hacerse completamente al principio, sino que podía hacerse a petición del simulador durante la ejecución.

A los personajes se les proporcionaban unos objetivos motivadores para que se produjeran los acontecimientos. También estaban definidos por un conjunto de habilidades, que indicaban al planificador qué tipo de acciones podía realizar cada uno. Roald también requería que el usuario especificara las características de cada personaje para que pudiera actuar de forma natural. Como muchos eventos de las historias implicaban la interacción entre dos personajes, también había que especificar la naturaleza de la relación entre los personajes.

A diferencia de Tale-Spin, las acciones previstas de un personaje en Roald se diferenciaban de los acontecimientos de la historia. Los planes de todos los personajes se enviaban al simulador para su ejecución. Mantener la planificación y la ejecución separadas permitía al simulador tener sus propios objetivos, lo que podía significar que los acontecimientos del mundo se desarrollaran de forma diferente a la voluntad de los personajes.

2.2.2.5. Universe

Universe (Lebowitz, 1984) fue diseñado para modelar la generación de guiones para una sucesión de episodios de telenovelas en los que un gran elenco de personajes puede representar múltiples historias simultáneas y superpuestas que nunca terminan. El interés de Universe radica en que fue el primer sistema de narración que dio especial importancia a la creación de personajes. Universe utilizaba complejas estructuras de datos para modelar los personajes, utilizando como entrada tanto estereotipos predefinidos como caracterizaciones proporcionadas por el usuario.

A diferencia de Dehn, que consideraba que la trama debía impulsar la creación del escenario y los personajes (Dehn, 1981a), Lebowitz desarrolló Universe considerando que los personajes podían crearse independientemente de la trama (Lebowitz, 1984). Debido a este enfoque, Universe podía generar historias que no tienen un final claro, y elegir entre continuar una historia anterior o desarrollar una nueva a partir de los objetivos y acontecimientos de los personajes.

2.2.2.6. GESTER

GESTER, *GEnerating STories from Epic Rules*, (Pemberton, 1989) fue una de las primeras aproximaciones a la generación de historias a partir de módulos de conocimiento independiente que interactuaban entre sí para dar contexto a la narración. El programa era un sistema de generación de historias basado en reglas, que manejaba información sobre la estructura de la historia en forma de una versión simplificada de una gramática narrativa. Esta aproximación se completaba con cono-

cimiento relativo a los posibles eventos y actores arquetípicos que pueden concurrir en el subgénero épico.

2.2.2.7. Tailor

Tailor (Smith y Witten, 1991) era un sistema que generaba historias de forma similar a Tale-Spin, es decir, construyendo y ejecutando planes para un conjunto de personajes dentro de una simulación del mundo. Así, cada situación estaba representada por un conjunto de hechos, y las acciones estaban representadas por sus precondiciones y efectos sobre ese conjunto de hechos.

La estrategia de planificación de Tailor seguía un encadenamiento hacia delante a partir de la situación actual, probando todas las acciones posibles para evaluar las situaciones resultantes mediante una función de puntuación. Mediante este enfoque, la función de evaluación podía medir qué cadenas acercarían a un personaje a satisfacer su objetivo.

Tailor también introdujo un segundo personaje que actuaba como antagonista del protagonista, tratando de impedirle alcanzar sus objetivos. Cada personaje se turnaba durante la generación de la historia, lo que daba al proceso de generación cierta dinámica de juego.

2.2.2.8. Minstrel

Minstrel (Turner, 1993) era un sistema generador que producía historias sobre el Rey Arturo y sus Caballeros de la Tabla Redonda. Todas las historias se centraban en una moraleja, que también proporcionaba la semilla para desarrollar cada una.

Las unidades de construcción que empleaba Minstrel eran una colección de objetivos y los correspondientes planes para satisfacerlos. También es una particularidad de Minstrel que diferenciaba dos niveles de objetivos: los objetivos del autor y los objetivos del personaje. La construcción de historias en Minstrel funcionaba como un proceso de dos etapas que incluía una etapa de planificación y una etapa de resolución de problemas.

Minstrel aplicaba un enfoque de razonamiento basado en casos para crear nuevas historias. También reutilizaba pasajes de historias generadas anteriormente tomados de una memoria episódica durante la fase de resolución de problemas (Malkewitz y Iurgel, 2006). Minstrel utilizaba una base de datos para apoyar este proceso. Dicha base de datos contenía esquemas de historias que incluían los objetivos de los personajes, las acciones y los estados de los objetos, junto con el conocimiento de los objetivos del escritor dramático.

A partir de Minstrel, Tearse et al. (2014) desarrollarían más adelante **Skald**, una reconstrucción del sistema tomando y ampliando los conceptos fundamentales considerados en el sistema.

2.2.3. Los generadores contemporáneos (1999-2022)

En esta categoría se recogen los sistemas generadores más modernos. Estos han aprovechado y combinado las estrategias y modelos desarrollados por los sistemas

clásicos, sin perjuicio de haber desarrollado también sus propios planteamientos innovadores. Algunos de los sistemas referidos en la sección destacan por su complejidad arquitectónico, así como por su capacidad de generar historias ricas y elaboradas.

2.2.3.1. Mexica

Mexica (Perez y Perez, 1999) es un sistema cuyo objetivo es estudiar el proceso creativo. Ha sido diseñado para generar historias cortas sobre los primeros habitantes de México (los mexicas, de ahí su propio nombre). Mexica ha sido un sistema generador pionero en la inclusión de los vínculos emocionales y las tensiones entre los personajes como medio para dirigir la creación y evaluar las historias en curso.

La arquitectura de Mexica se basa en una secuencia de dos procesos principales: uno que crea todas las estructuras de datos en la memoria a largo plazo, y un segundo proceso que genera las nuevas historias utilizando estas estructuras.

Mexica utiliza varias estructuras de conocimiento para apoyar su modelo de narración: Una biblioteca de acciones, una colección de historias para inspirar la generación de nuevas historias, y un conjunto de personajes y localizaciones (Perez y Perez, 1999).

Mexica no sólo emplea historias generadas para inspirar otras, también procesaba historias creadas por autores humanos. Estas historias se incluyen en el proceso como una historia más de las generadas por Mexica, como secuencias de acción. Así, cada historia inspiradora se analiza para utilizarla como parte del conocimiento contextual.

2.2.3.2. Brutus

Brutus (Bringsjord y Ferrucci, 1999), es un sistema que genera historias cortas utilizando la traición como leitmotiv. La principal aportación de Brutus ha sido su rico modelo lógico para representar la traición. Esta característica le ha permitido generar historias bastante complejas. Otro aspecto innovador de Brutus es que se diseñó teniendo en cuenta el conjunto de conocimientos existentes sobre literatura y gramática. Con esta combinación, Brutus ha sido capaz de crear historias realmente notables, muy cercanas a la producción literaria humana.

Brutus maneja conocimientos específicos para crear un escenario adecuado para que las historias tengan lugar. Además, estructura el conocimiento en varias capas: conocimiento del dominio, conocimiento lingüístico, conocimiento literario (que contiene información para mejorar la calidad literaria) y gramáticas literarias aumentadas. Esta combinación de niveles de conocimiento es clave para la calidad de las historias generadas.

2.2.3.3. MakeBelieve

MakeBelieve (Liu y Singh, 2002), era un sistema de generación de historias cortas de ficción que utilizaba conocimiento de sentido común para generar historias. Básicamente, el usuario proporcionaba una historia sobre un personaje como semilla inicial y, a continuación, MakeBelieve intentaba continuar esa historia infiriendo posibles secuencias de eventos que pudieran ocurrirle a ese personaje. El

sistema utilizaba conocimientos de sentido común sobre la causalidad y el funcionamiento del mundo, extraídos de la base de conocimientos Open Mind Common Sense (Singh et al., 2002a). Para generar el producto final, combinaba este conocimiento con técnicas lingüísticas de generación de historias. MakeBelieve también utilizba el sentido común para realizar una evaluación crítica de la historia generada, con el fin de encontrar pasajes incoherentes o poco sólidos desde el punto de vista del sentido común.

2.2.3.4. Hefti

Hefti (Ong y Leggett, 2004) es un caso destacable de aplicación de algoritmos genéticos para la generación de historias. En este sistema, el autor debe especificar una estructura básica, o "hilo argumental", que debe cumplir la trama. La idoneidad de una historia viene determinada por los eventos, cada uno de los cuales debe ser calificado por el autor en función de su agrado/desagrado por un determinado evento. Así, el algoritmo genético trata de obtener el mejor resultado posible según las valoraciones del autor, asegurando al mismo tiempo que se cumplan los requisitos del hilo argumental de la historia. Este algoritmo genético también se utiliza para modificar la trama si hay desviaciones de los participantes.

2.2.3.5. Virtual Storyteller

Virtual Storyteller (Faas, 2002; Swartjes, 2006) es un sistema multiagente que puede generar historias simulando un mundo virtual en el que los personajes modelados por agentes persiguen sus objetivos. De este modo, la historia emerge de los acontecimientos del mundo virtual.

Virtual Storyteller ha ido evolucionando con el tiempo, añadiendo nuevas características y superando así su alcance original. Una de sus características más interesantes, añadida recientemente (Brinke, 2014), es el conjunto de reglas de visibilidad que el agente del personaje aplica a las percepciones entrantes. El agente sigue manteniendo en la memoria una versión completa y precisa del mundo de la historia que el agente puede utilizar a nivel de actor. Esto significa que los personajes utilizan una versión incompleta y posiblemente incorrecta del mundo de la historia para hacer planes. Para ayudar al personaje a hacer planes en caso de que no disponga de todo el conocimiento necesario, se introduce un operador de suposición. Esto permite al personaje hacer suposiciones sobre la ubicación de los objetos y decidir si un objetivo es o no alcanzable, lo que le permite buscar objetos.

Virtual Storyteller estructura sus conocimientos en cinco niveles. Los dos primeros niveles se centran en la descripción de los elementos básicos de la historia, como los personajes, los objetos y el entorno implicados en la historia con sus propiedades y relaciones. El tercer nivel representaba la trama de la historia, entendida como la semántica de las historias, captada mediante la vinculación de los elementos de la historia con las acciones, los acontecimientos y los procesos de fondo. Además, los dos últimos niveles incluían el metaconocimiento sobre la estructura de la historia, es decir, los conceptos narratológicos y la información del usuario sobre el impacto emocional.

2.2.3.6. Fabulist

Fabulist (Riedl y Young, 2010) es una arquitectura completa para la generación y presentación automática de historias. Fabulist combina un enfoque centrado en el autor junto con una representación de la intencionalidad de los personajes y una planificación de mundo abierto para maximizar la coherencia y la credibilidad de la historia.

El proceso de generación narrativa de Fabulist se estructura en tres niveles: generación de fábulas, generación de discursos y representación de medios. El proceso de generación de fábulas utiliza un enfoque de planificación para la generación narrativa. En el caso de Fabulist, las entradas proporcionadas incluyen un modelo de dominio que describe el estado inicial del mundo de la historia, las posibles operaciones que pueden realizar los personajes y un resultado.

El sistema Fabulist modela el proceso de generación a través de una extensión del paradigma de planificación de enlace causal de orden parcial (POCL), la planificación de historias impulsada por la intención (IPOCL, Intent-Driven Partial-Order Causal Link). La planificación de orden parcial retrasa su curso de decisión sobre el orden de las acciones para dejarlo lo más abierto posible. Contrasta con la planificación de orden total, que produce un ordenamiento exacto de las acciones. Dado un contexto en el que se requiere alguna secuencia de acciones para lograr un objetivo, un plan de orden parcial especifica todas las acciones que deben realizarse, pero sólo especifica un orden de las acciones cuando es necesario.

2.2.3.7. Thespian

Thespian (Si, 2010) aporta un nuevo enfoque en la emulación de la subjetividad de los personajes al modelar sus creencias. Proporciona a los personajes una representación de sus propias creencias sobre el mundo con el que interactúan, y esas creencias pueden ser incluso falsas. Cada personaje tiene su propia representación del mundo, incluyendo sus propias creencias, y conocimiento sobre otros personajes para inferir cómo ven el mundo.Esta característica permitía a los agentes simular la subjetividad de los demás y razonar sobre sus emociones, sus creencias y su comportamiento social.

2.2.3.8. Scheherazade

Scheherazade (Li et al., 2012, 2013) utiliza el conocimiento minado a partir de aportaciones masivas individuales, *crowd-sourced knowledge*, para adquirir automáticamente el conocimiento del dominio necesario para construir y comprender historias sobre actividades cotidianas, como ir a un restaurante o ir al cine. Además de la parte de generación narrativa, está es el sistema Scheherazade-IF (Li et al., 2012), que crea experiencias narrativas interactivas al permitir que un usuario humano asuma el papel de uno de los personajes del dominio.

2.2.3.9. Slant

Slant (Montfort et al., 2013), es un sistema para la generación de historias creativas que integra diferentes tipos de experiencia y creatividad. Slant también proporciona un framework práctico con vistas a que otros sistemas se integren en él.

Slant es el resultado final de un ambicioso proyecto de integración en el que han participado varios sistemas de narración existentes. Slant emplea un arquitectura de pizarra, o blackboard (Hayes-Roth, 1985), para permitir la colaboración de varios componentes diferentes en el proceso de generación de historias: uno basado en Mexica (Perez y Perez, 1999), otro basado en GRIOT, y uno nuevo desarrollado específicamente para Slant. GRIOT (Harrell, 2006), es un sistema desarrollado para implementar sistemas que producen narrativas interactivas.

2.2.3.10. STellA

STellA (Storytelling Algorithm) (León y Gervás, 2014) es un sistema de generación de historias que mezcla una producción de estados del mundo basada en la simulación no restringida y acciones narrativas como material fuente para un motor de exploración espacial conceptual. El sistema gestiona los estados en un espacio generado de forma no determinista de historias parciales, haciendo elecciones hasta que encuentra una simulación satisfactoria de la progresión de eventos de las simulaciones que se representa como una historia.

En STellA, el proceso de generación implica una creación iterativa de nuevos estados. Cada simulación se modela e implementa como un proceso no determinista en el que cada paso puede generar no uno sino muchos pasos. Esta simulación requiere que todo el dominio del mundo se represente explícitamente como una visión simplista de un entorno realista. Este enfoque proporciona un escenario muy detallado que permite un rico conjunto de posibilidades en la generación. Cada iteración genera un conjunto de versiones candidatas del estado actual y, a continuación, el proceso identifica las más probables analizando su verosimilitud en términos de su plausibilidad y sus propiedades narrativas. Este paso se lleva a cabo aplicando restricciones y una versión generalizada de las curvas de tensión para impulsar la generación de historias. Estas historias parciales candidatas se evalúan en la medida en que satisfacen un conjunto determinado de restricciones y en qué medida sus curvas de tensión se ajustan a un conjunto de curvas objetivo. Los resultados de este proceso proporcionan un criterio para decidir si una historia parcial es prometedora y si una historia está terminada.

El conocimiento subyacente que impulsa la generación tiene un gran impacto en la calidad final del resultado en STellA. Puede decirse que se basa en gran medida en su conjunto de reglas básicas, utilizadas para representar el universo en el que se desarrollan sus historias.

2.2.3.11. Charade

Charade (Méndez et al., 2014) es un sistema de generación de historias basado en afinidades que genera historias tras simular la evolución de una relación entre dos

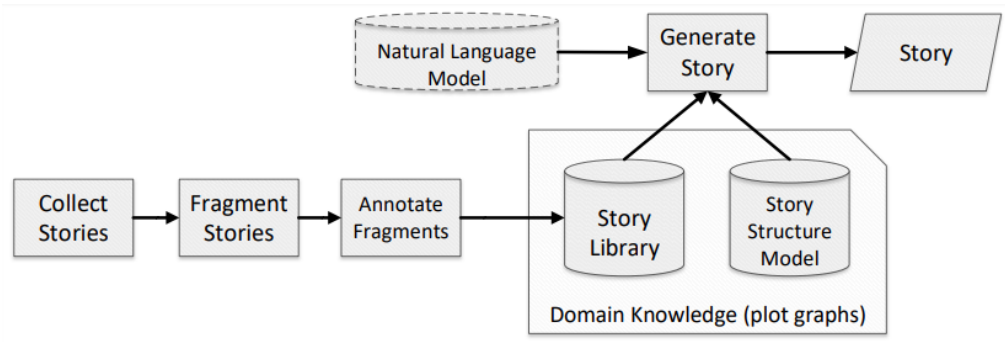


Figura 2.1: Framework para un modelo estructural de generación de historias. Fuente: Alhussain y Azmi (2021).

personajes utilizando sus afinidades mutuas.

Este sistema es una arquitectura basada en agentes desarrollada con JADE y consta de dos tipos de agentes: un Agente Director, que configura el entorno de ejecución y crea los personajes; y los Agentes Personales, uno por cada personaje de la historia, cuyas interacciones generan la historia.

El objetivo principal del sistema era implementar un modelo de afinidad lo más desvinculado posible del dominio de la historia, y probarlo independientemente de otros factores como el entorno en el que se desarrolla la acción o los rasgos de personalidad y el estado emocional de los personajes. Gracias a esta independencia, puede utilizarse fácilmente para generar distintos tipos de historias.

2.3. Estrategias de generación

Esta sección analiza las diversas estrategias de generación de historias aplicadas por los diferentes sistemas mencionados; y, en algunos casos, las implicaciones de dichas estrategias en la arquitectura subyacente, con un especial interés en los sistemas que generan historias con tramas múltiples.

El modelo general de generación queda muy bien sintetizado en la Figura 2.1, tomada del trabajo de Alhussain y Azmi (2021).

Alhussain y Azmi (2021) clasifican las diversas estrategias de generación en estructurales, basadas en planificación, basadas en la analogía, basadas en búsqueda heurística y finalmente, en aprendizaje automático (Machine Learning).

2.3.1. Estrategias de generación estructurales

Uno de los primeros y más extendidos esfuerzos por formalizar las historias en modelos estructurales es el trabajo del formalista ruso Vladimir Propp. En su libro "Morfología del cuento" (Propp, 1968), llegó a la conclusión de que todos los cuentos populares se componen de las mismas treinta y una acciones de los personajes, a las que denominó *funciones*, como *ausencia*, *villanía*, *carencia*, *lucha* y *victoria*. Estas funciones pueden no aparecer en todos los cuentos; sin embargo, las funciones que

aparecen en un cuento siguen un orden prefijado e invariable. De este enfoque surgió el concepto de gramática de historia, una aproximación que se describe en esta sección y que fue estudiado en profundidad por Rumelhart (1975b) y Thorndyke (1977a).

Otro paso en esta dirección viene de los trabajos de Schank y Abelson (1977, 2013), las teorías de la gramática de las historias consideran estas últimas como guiones (*script*), donde un guión es una estructura que describe una secuencia adecuada de acontecimientos en un contexto determinado. Estas teorías surgieron del hecho de que, en el mundo real, los acontecimientos suelen producirse siguiendo patrones estereotipados Schank y Abelson (1977). Por ejemplo, cuando alguien quiere comer en un restaurante, la secuencia de acontecimientos es la siguiente: entrar en el restaurante, sentarse a la mesa, leer el menú, pedir comida, comer, pagar el dinero y salir del restaurante. Estas secuencias, o patrones, se utilizan como esquemas para guiar la generación de historias. Sin embargo, un contexto puede tener diferentes escenarios que varían ligeramente, lo que da lugar a diferentes variaciones. Por ejemplo, si no hay menú sobre la mesa, el cliente deberá pedírselo al camarero. Estas variaciones permiten que haya mayor variabilidad en las historias generadas.

Más adelante, el propio concepto de gramática de historias sería cuestionado por Black y Wilensky (1979) y el debate en torno a ella se prolongó durante muchos años y condujo al descrédito de las gramáticas. Sin embargo, las gramáticas de historias aún gozan de cierta popularidad, y en su momento, los sistemas basados en estas produjeron con éxito un número de historias de alta calidad. En este sentido el concepto de gramática de historias para la generación de historias puede considerarse validado como una tecnología sólida y exitosa.

2.3.1.1. Estrategias basadas en grafos

La forma más sencilla de un guión utilizado en la generación de historias es la construcción de grafos de historias. Durante la fase de diseño, se construye un grafo de historias. Se trata de una estructura que representa el espacio de todas las historias posibles, con ramas para cada variación surgida a partir de un estado del universo de la historia. Luego, en la fase de generación, se recorre el grafo para encontrar un camino lineal que representa una historia generada. La calidad de la historia generada depende principalmente de la calidad del grafo construido y del algoritmo de recorrido. Mediante la introducción de restricciones en la búsqueda del grafo se pueden obtener diversas soluciones y se puede mejorar los resultados.

Basándose en la estructura de cuentos de Propp, Maranda (1985) desarrolló un grafo que permite generar cuentos populares. El grafo de Maranda consiste en nodos que contienen las funciones de Propp con nodos de inicio y nodos de terminación. En la base de conocimientos anotada de cuentos populares rusos se busca un fragmento que que coincida con la función del nodo en cada nodo atravesado. El fragmento recuperado se concatena con la historia generada. El proceso se repite hasta llegar a un nodo terminal. Este método no sólo genera los cuentos populares rusos originales, sino que también puede generar nuevos cuentos. Sin embargo, el grafo de Maranda es cíclico, lo que puede llevar al sistema a infinitas iteraciones.

El sistema Scheherazade (Li et al., 2013; Li y Riedl, 2015) recoge las experiencias

humanas sobre un dominio temático en forma de guiones. A continuación, aprende un grafo argumental basado en estos guiones. El grafo se recorre para generar historias. Los grafos de Scheherazade son similares al grafo de Maranda, ya que ambos tienen nodos de inicio y finalización específicos. Sin embargo, en el último caso, los grafos son acíclicos y se aplican restricciones mutuas en algunos de los eventos de las historias generadas.

2.3.1.2. Estrategias basadas en gramáticas

El uso de gramáticas para generar historias comenzó cuando Lakoff (1972) reformuló la estructura de historias de Propp en una gramática del relato. Considerando las historias como palabras del lenguaje formal de la narrativa, donde los de Propp, Lakoff utilizó reglas de reescritura ampliables para generar historias, lo que que permitían producir diferentes historias seleccionando distintas expansiones. Su trabajo inspiró a investigadores a proponer otras gramáticas de historias. Pemberton (1986) propuso una gramática de historias para una antigua epopeya francesa, que se implementó posteriormente como GESTER, (Pemberton, 1989), un programa que genera modelos de historias basados en la propia gramática propuesta por Pemberton. Sus historias tienen un principio claro medio y final. Brutus, desarrollado por Bringsjord y Ferrucci (2000), es también un sistema que genera historias de traición basadas en gramáticas de historias. Crea historias complejas basadas en estructuras de marco, donde cada elemento de la historia, como los personajes y los eventos, se consideran marcos de la historia. Estos marcos se agrupan en varios temas de la historia. Todas las gramáticas de historias mencionadas anteriormente son gramáticas especializadas limitadas al ámbito que las produce. Por lo tanto, sólo pueden generar un pequeño conjunto restringido de historias, por lo que es necesario recurrir a gramáticas de historias más generales.

Se suele conceder a Rumelhart (1975b) el mérito de haber sido, teóricamente, el primero en proponer una gramática general de la historia, entendida como narrativa. Desde entonces, se han propuesto varias gramáticas generales de historias, incluida la de Thorndyke (1977a), que fue ampliamente adoptada debido a su simplicidad. En conclusión, los modelos basados en gramáticas de generación de historias son enfoques fáciles de implementar, que permiten una generación rapida en términos de tiempo de ejecución, y que permiten generar historias bien estructuradas, siempre que el modelo estructural esté bien articulado. También pueden generar historias interesantes. No obstante, como ya se ha dicho, los modelos estructurales se centran en la estructura de la historia; es decir, se centran en la sintaxis de la historia y no en su semántica. Y es preciso indicar que las historias son modelos semánticos por naturaleza. Por lo tanto, las relaciones lógicas entre los acontecimientos de la historia y entre las intenciones y acciones de los personajes se verán afectadas negativamente. Esta limitación en el plano semántico afecta a la coherencia y la credibilidad de la historia. Además, los modelos estructurales son rígidos; sólo pueden generar historias que satisfacen la estructura de la historia proporcionada y no pueden modificar sus conocimientos para generar historias diferentes. Las historias generadas por este mecanismo se limitan a un solo protagonista porque tener más de de un protagonista requiere relaciones

lógicas complejas. Por último, la generación de estructuras de historias sufre del problema de la sobregeneración. Es decir, el sistema puede generar textos que no son historias y aceptarlos como salidas válidas (Alhussain y Azmi, 2021).

Las teorías de la gramática de historias fueron criticadas con un enfoque de semántica de la historia (Black y Wilensky, 1979; Black y Bower, 1980). Según los psicólogos, las gramáticas de las historias pueden construir una historia sintácticamente. Sin embargo, no tienen en cuenta las relaciones semánticas de la historia. Por lo tanto, no pueden aplicarse a historias con objetivos contradictorios o con múltiples protagonistas. El peor caso de las gramáticas de historias es cuando aceptan las no-historias como historias (Black y Bower, 1980). La teoría de los puntos de la historia se propuso como respuesta a estas críticas. De acuerdo con este enfoque, se plantea una historia como una cadena de acontecimientos conectados causalmente para perseguir un objetivo final, (Wilensky, 1983). La conexión causal entre los acontecimientos tiene más sentido para el lector y sirve a la semántica de la historia. Esta aproximación ha influido fuertemente en otro mecanismo que se describe a continuación, la generación automática de historias mediante algoritmos de planificación de Inteligencia Artificial. Sabiendo que tanto la planificación teórica de historias como la planificación de IA se basan en el razonamiento la analogía entre ambos parece evidente. En general, la generación de historias mediante el uso de planificadores se base en definir un estado inicial y un objetivo, para que un razonador infiera qué acciones se pueden y se deben realizar para pasar de un estado a otro y, en última instancia, lleve el estado inicial al objetivo a través de estados intermedios (Alhussain y Azmi, 2021).

2.3.2. Estrategias basadas en la planificación

Estos fueron los primeros generadores de historias inteligentes, seguidos de los modelos estructurales. Con los agentes basados en objetivos, que van desde simples agentes atómicos de resolución de problemas hasta agentes estructurados de planificación, se utilizaron planificadores de historias en una amplia gama de generadores de historias en la literatura.

Meehan (1977) fue el primero en introducir la planificación para la generación automática de historias. A diferencia de las gramáticas de historias, Tale-Spin se concentraba en las necesidades de los personajes y en sus intenciones de satisfacer dichas necesidades utilizando técnicas de resolución de problemas clásicas en Inteligencia Artificial. A través de la construcción de un planificador de simulación del mundo, Tale-Spin se guiaba totalmente por los objetivos de los personajes. La generación de la historia comienza estableciendo un estado inicial, es decir, una descripción del mundo de la historia y uno o más objetivos de los personajes. A continuación, utilizando un motor de inferencia (razonador) para implementar un algoritmo de de encadenamiento, el plan de la historia se produce mediante la inferencia de cadenas de eventos causales considerando el efecto de cada evento en el mundo de la historia. El proceso continúa hasta que se alcanza el objetivo del personaje.

Este enfoque proporciona a los personajes de la historia intenciones claras y, por tanto, mejora su credibilidad. Tale-Spin fue capaz de generar historias cortas y

coherentes similares a las fábulas de Esopo. Para una historia bien estructurada, hay que declarar el estado inicial del mundo y los objetivos de los personajes. Sin embargo, centrarse en las necesidades de un personaje y en sus acciones para satisfacerlas puede dar lugar a historias poco interesantes que carecen de clímax o resolución. Otro defecto de Tale-Spin es que muchas de sus buenas historias generadas son un conjunto de las historias originales utilizadas para construir su rígida base de conocimientos.

Riedl y Young (2010) propusieron Fabulist, un planificador de Orden Causal Parcial Dirigido por la Intención (IPOCL) que consta de dos mecanismos: el primero es el planificador de enlaces causales de orden parcial que infiere cadenas de acciones causales de los personajes impulsadas por los objetivos globales del autor. El segundo mecanismo pretende preservar la credibilidad de los personajes simulando el proceso de reconocimiento del público (Carberry, 2001). Este mecanismo es un proceso de razonamiento único integrado en el planificador que toma las acciones de los personajes y trata de predecir la intención del personaje (objetivo) basándose en estas acciones. Si la meta del personaje no es predecible, las acciones del personaje no se considerarán intencionadas y, por tanto, el plan se considerará defectuoso y tendrá que ser revisado para garantizar la credibilidad del personaje.

Sin embargo, los objetivos muy predecibles darán lugar a una historia poco interesante. El planificador IPOCL es lento, puede tomar varias horas en generar un plan completo (Riedl y Young, 2010). Además, el hecho de que IPOCL utilice un lenguaje de representación no estándar limita sus mejoras basadas en planificadores estándar que tienen un rendimiento más rápido. Para abordar la planificación narrativa orientada a la intención mediante planificadores clásicos en lugar del planificador especializado IPOCL, Haslum (2012) remodeló el problema de planificación narrativa para incorporar la intencionalidad de los personajes. Modeló las intenciones de los personajes como parte de la especificación del problema de planificación narrativa. Luego, utilizó las intenciones como precondiciones de las acciones de los personajes. Esta compilación permite el uso de planificadores estándar para generar historias y acelerar el proceso de generación. Otra extensión de IPOCL es el algoritmo de planificación de enlaces causales de orden parcial de conflicto (CPOCL) propuesto en Ware y Young (2011). CPOCL crea un modelo de conflicto y luego impone la generación de conflictos en las historias restringiendo el planificador. Esto se hace mediante el uso de pasos no ejecutados para modelar las intenciones frustradas de los personajes que permiten la ejecución parcial de los planes.

2.3.2.1. Estrategias basadas en la simulación

En el enfoque basado en la simulación, los sistemas generadores permiten la creación de universos o espacios de interacción para los personajes de la historia. A partir de la libre interacción entre los personajes, se van sucediendo acciones y eventos, cambios en el propio universo de la historia, y una modificación de los parámetros relativos a la relación entre los personajes. El enfoque simulativo es muy popular en la Narrativa Interactiva, precisamente por su proximidad al mundo de los videojuegos.

En el campo de la generación automática, dos buenos ejemplos de este modelo de generación son Charade y STellA. En el caso de Charade (Méndez et al., 2016), el espacio de la historia está formado por una serie de personajes que guardan unas relaciones (de amistad, amor o enemistad) y que actúan conforme a la intensidad de sus relación con el otro personaje. A partir de una simulación libre, se obtiene una serie de acciones y una secuencia de cambios en los parámetros de la relación entre los personajes.

STellA (León y Gervás, 2014) es un sistema aún más complejo, donde la simulación alcanza no sólo a los personajes, sino a todo el entorno en el que interactúan. STellA posee sistemas de reglas relativos a la física del espacio donde tiene lugar la acción (leyes físicas como la Gravedad), así como las normas de comportamiento social aceptable, etc. Es la expresión de la generación simulativa llevada a todos los niveles.

2.3.2.2. Estrategias de esquema global

Esta estrategia se puede considerar como una forma de emular el trabajo de los autores literarios, cuando escriben historias que persiguen ser coherentes con respecto a un esquema previo. Para ello, parten de la creación de un conjunto de objetivos que forman el esqueleto de la historia; luego, dirigen a los personajes de la historia para que persigan esos objetivos. Para ajustarse al esquema de la historia, los sistemas de generación que siguen este enfoque han simulado este proceso desplazando la importancia de la consecución de los objetivos de los personajes a la consecución de los objetivos del autor (correspondientes a un esquema global que debe cumplir la historia). Los objetivos perseguidos por el autor son independientes de los perseguidos por los personajes de la historia, lo que puede dar lugar a una competencia o a un conflicto de objetivos, lo que supuestamente aumentaría el interés de la misma.

Dehn (1981a) propuso un enfoque basado en esquema global para Author. Este sistema generador implementa una arquitectura de memoria dinámica reconstructiva que simula a un autor humano que escribe el borrador de una historia y lo revisa varias veces antes de entregarlo. El proceso de generación de historias comienza con un conjunto de objetivos de autor que se proporcionan como entrada. A continuación, comienza un bucle de tres subtareas. En primer lugar, se inicia la búsqueda de materiales relacionados en la memoria. A continuación, se selecciona la parte más adecuada de los materiales recuperados. Por último, la reformulación conceptual tiene lugar revisando los objetivos del autor y modificándolos si es necesario. El bucle continúa hasta que se satisfacen todos los objetivos del autor. Las historias generadas por Author están generalmente bien estructuradas y son más interesantes que las generadas por un sistema basado en los objetivos de los personajes. Sin embargo, la credibilidad de los personajes se vio afectada negativamente porque (los personajes) a veces actuaban sin intenciones claras para satisfacer los objetivos del autor.

2.3.2.3. Estrategias multiagente

Con el fin de generar historias coherentes y bien estructuradas, los investigadores han intentado dirigir la generación de historias tanto por los objetivos de los

personajes como por los del autor. Este enfoque es similar al de los objetivos de los personajes, ya que éstos son dirigidos por sus propios objetivos, lo que preserva la coherencia de la historia. Virtual Storyteller (Faas, 2002; Swartjes, 2006) es un sistema de generación que utiliza agentes agentes inteligentes para generar historias. Mientras los personajes planifican para alcanzar sus propios objetivos, un agente director virtual que tiene conocimientos generales sobre la estructura de la trama se propone dirigir las acciones de los personajes para preservar la estructura simple de la historia: un principio, un medio y un final feliz. Esto se consigue dirigiendo la trama en la dirección deseada utilizando un control ambiental, por ejemplo, introduciendo nuevos personajes; y un control motivacional, por ejemplo, introduciendo nuevos objetivos de los personajes.

Otro ejemplo de planteamiento multiagente lo tenemos en Charade (Méndez et al., 2014), un sistema generador que empleaba agentes para representar a los personajes y el intercambio de mensajes entre ellos, como mecanismo de emulación de interacciones entre los personajes y la evolución de sus relaciones.

2.3.3. Estrategias basadas en la analogía

La analogía en computación es un enfoque de la IA que se basa en el proceso cognitivo humano de razonamiento analógico. El razonamiento analógico humano consiste en obtener una conclusión a partir de premisas en las que se establece una similaridad o analogía entre elementos o conjuntos de elementos distintos. Su traducción computacional opera identificando similitudes y transfiriendo conocimientos entre un dominio de origen y un dominio de destino (Zhu y Ontañón, 2013). Utilizando la analogía, se puede resolver un nuevo problema aplicando la solución a un problema similar previamente conocido. Este enfoque se aplica en los sistemas de generación de historias mediante la búsqueda en la base de conocimientos un estado del mundo de la historia similar al estado del mundo de la historia actual. Una vez localizado, se obtiene el siguiente evento de la historia que se está generando a partir del que se daría en el mundo análogo. La medida de similitud necesaria para establecer esta analogía difiere entre los distintos sistemas generadores.

Minstrel (Turner, 1993) es uno de los primeros sistemas basados en la analogía para la generación de historias. Es un sistema complejo impulsado por los objetivos del personaje y del autor, en el que el razonamiento basado en casos (CBR) se utiliza principalmente para lograr los objetivos del personaje. Minstrel almacena las escenas (casos) en su memoria episódica, donde las escenas se indexan por medio de pistas destacadas, como la ubicación y la acción, para formar grupos de escenas relacionadas. Cuando la instanciación de los esquemas temáticos de la historia falla porque no hay escenas que coincidan en la memoria, Minstrel crea nuevas escenas mediante el uso de métodos de adaptación de la recuperación de la transformación (TRAMS). En primer lugar, para simplificar la búsqueda de escenas similares en la memoria episódica, las especificaciones del esquema de la historia se transforman en una forma general sustituyendo actores y objetos por “alguien” y “algo”, respectivamente. Después de recordar escenas similares de la memoria episódica, las escenas recuperadas se adaptan para que coincidan con la especificación original de la histo-

ria. Cabe destacar que Minstrel evita utilizar una escena más de más de dos veces para garantizar la novedad de la historia.

Mexica (Perez y Perez, 1999) crea un contexto de historia-mundo para registrar los vínculos emocionales con otros personajes y las tensiones dramáticas producidas en la historia. Estos elementos tácitos funcionan como condiciones previas y posteriores a las acciones de la historia. En su memoria a largo plazo (LTM), Mexica almacena diferentes esquemas de elementos tácitos, y cada esquema está asociado a un conjunto de posibles acciones posteriores obtenidas mediante el análisis de historias anteriores. Durante el proceso de compromiso, se busca en la MLP un esquema que coincida con alguno de los contextos del mundo de las historias. Si se encuentra un esquema que coincida, se seleccionará una de las acciones asociadas a él como la siguiente acción de la historia, y los contextos del mundo de las historias se actualizarán en función de la acción seleccionada. Durante el proceso de reflexión, Mexica revisa la historia generada y evalúa su consistencia, novedad e interés en comparación con historias anteriores. Si estos requisitos no se cumplen, se producirán directrices que funcionarán como filtros de acciones cuando el proceso de compromiso se inicie de nuevo.

2.3.4. Estrategias de búsqueda heurística

Para aumentar la variedad de historias generadas, los investigadores han ampliado el dominio de búsqueda de historias. Sin embargo, a medida que el espacio de búsqueda crece, las técnicas de planificación tradicionales tienen dificultades para encontrar una solución eficaz. Por ello, se han introducido técnicas de búsqueda heurística. Hefti (Ong y Leggett, 2004) utiliza Algoritmos Genéticos para generar historias dividiendo cada historia en pasos de tiempo que son representados por componentes de la historia. Cada componente de la historia se codifica en un cromosoma en el que los genes son elementos de la historia, incluyendo agentes, eventos y objetos. La aptitud de un cromosoma se calcula sumando el atributo de aptitud de cada uno de sus elementos de la historia que son evaluados manualmente por los autores.

McIntyre y Lapata (2010) aplicaron algoritmos genéticos para generar historias. Primero extrajeron grafos argumentales de un corpus de historias donde cada nodo representa un único evento de la historia asociado a sus argumentos, como sustantivos, adverbios o adjetivos. A continuación, se creó una población inicial de historias mediante el muestreo del grafo de historias. Para generar nuevos cromosomas, se aplica un cruce de un punto. La mutación se aplica sustituyendo un evento de la historia, es decir, un nodo, o sustituyendo uno de los argumentos del evento por un argumento semánticamente similar. La aptitud de los cromosomas se basa en su coherencia, que se calcula utilizando la medida de representación de documentos de la red de entidades para la coherencia local (Barzilay y Lapata, 2008). Kartal et al. (2014) formularon la generación de historias como un problema de búsqueda de Árbol Monte Carlo. Cada nodo del árbol representa un estado de la historia, y cada arista representa una acción que que cambia un estado a un posible estado sucesivo. La historia mejor generada se elige en función de una función de evaluación que tiene

en cuenta dos factores: el porcentaje de objetivos del usuario logrados por la historia y el producto de la credibilidad de cada acción. Esta última es una medida definida por el usuario.

2.4. Modelos de representación del conocimiento

La representación del conocimiento es el formato y el método utilizados para codificar el conocimiento en la base de conocimientos de sistema computacional (Rusell y Norvig, 2003). Puede ser un lenguaje natural (Helbig, 2006), casos (Aamodt y Plaza, 1994), reglas (Rusell y Norvig, 2003), scripts (Schank y Abelson, 1977), frames (Minsky, 1975), redes neuronales (Hertz et al., 1990), redes semánticas (Helbig, 2006), etc. Diferentes representaciones del conocimiento y sus mecanismos de inferencia se han incorporado a los denominados sistemas basados en el conocimiento para resolver problemas complejos.

2.4.1. Representación basada en frames

Universe (Lebowitz, 1985a) introdujo el concepto de representación basada en frames de persona (*person frame*). Un frame es una colección de datos estructurados que describen a un personaje. El concepto de partida se basaba en desarrollos anteriores (Carbonell, 1979; Schank y Lebowitz, 1979). El enfoque de Universe se centra en la recopilación de rasgos de personalidad, relaciones interpersonales y, de alguna manera, objetivos del personaje. Sin embargo, el punto central de Universe era garantizar la coherencia de la narración. Esto significa que no toda la información relativa a un personaje es útil para desentrañar su comportamiento. Así, Lebowitz (1985a) distinguía dos categorías: los estereotipos y los acontecimientos pasados.

Los estereotipos son descripciones comunes que se asocian a personas de diversas clases, como ocupaciones, grupos sociales y antecedentes personales (Lebowitz, 1985a). A pesar de que los estereotipos proporcionan suficiente información de fondo para hacer ciertas suposiciones relacionadas con el modo de vida de los personajes, todo ello puede ser anulado. Los rasgos de personalidad desempeñan un papel en este sentido.

Los acontecimientos pasados se utilizan principalmente para aportar variedad y sabor, y explicar aspectos de un personaje que no se ajustan a los estereotipos, aunque es posible crear una gran variedad de personajes simplemente combinando estereotipos (Lebowitz, 1985a). Este mecanismo se introdujo para generar variaciones sobre los arquetipos de personajes que, de otro modo, habrían sido insustanciales.

Por último, el resto del conocimiento representado se refiere a las relaciones personales de los personajes. Debido a que los acontecimientos de la historia pueden suponer cambios en las relaciones, es necesario representar cada relación entre cada par de personajes. Universe destacaba las relaciones familiares como un tipo especial de relaciones para tratarlas de forma específica.

Así pues, cada frame está compuesto por (Lebowitz, 1985a):

- Nombre

- Estereotipos
- Modficaciones de los rasgos
- Objetivos individuales
- Relaciones interpersonales
- Matrimonio
- Historia (lista de eventos pasados)

Otro sistema que empleaba frames, en combinación con otros mecanismos, era Minstrel (Turner, 1993). La representación del conocimiento en Minstrel utilizaba una extensión de una biblioteca Lisp llamada Rhapsody (Malkewitz y Iurgel, 2006). Minstrel usaba frames, esquemas con ranuras y facetas que representan temas o moralejas de la historia, efectos dramáticos (suspense, presagio, ritmo, diálogo...), estados del mundo, creencias y afectos de los personajes, etc.

2.4.2. Representación basada en scripts

Un **script** (Schank y Abelson, 1975a) es una estructura que describe una secuencia apropiada de eventos en un contexto particular. Un script se compone de ranuras y requisitos sobre lo que puede llenar esas ranuras. Las ranuras no son estructuras aisladas, sino que el contenido de una sola ranura puede afectar a otra. Conceptualmente, los scripts modelan situaciones cotidianas, representando una especie de conocimiento común. En este sentido, un script es una secuencia predeterminada y estereotipada de acciones que definen una situación conocida (Schank y Abelson, 1975a).

Para cubrir aquellos casos en los que no se podía aplicar una situación estereotipada, algunos sistemas hacían uso de un constructo denominado plan. Un **plan** (Schank y Abelson, 1975a) representa el conocimiento de conjuntos de acciones necesarias para lograr determinados objetivos y se utiliza en situaciones no estereotipadas en las que no hay un script disponible. Los planes son responsables del comportamiento deliberado que muestran las personas. Los planes describen el conjunto de opciones que tiene una persona cuando se dispone a cumplir un objetivo.

Un ejemplo de sistema que hacía uso de este modelo es Tale-Spin (Meehan, 1977). La representación del conocimiento de Tale-Spin se basa en la Teoría de la Dependencia Conceptual (Schank y Abelson, 1975b; Schank, 1975). El resultado de su proceso generativo puede considerarse como un registro a través del proceso de resolución de problemas. Estos problemas se limitan a un área específica de conocimiento, el dominio del problema. Este dominio de problemas particular está definido por un conjunto de primitivas, un conjunto de estados de meta o problemas, y procedimientos para lograr estas metas. La base teórica de esta representación procede de la teoría de los esquemas, y más concretamente de (Schank y Abelson, 1975a).

Otro sistema con un mecanismo de representación fuertemente influenciado por los trabajos de Schank es Author (Dehn, 1981a). Como este sistema trataba de emular

el proceso creativo que sigue la mente humana la representación del conocimiento en Author se basa en una memoria estructurada y poblada de forma similar a la humana. Esta memoria debe contener tanto hechos relativos al mundo en el que se desarrolla la acción, como recuerdos de episodios notables, personajes y cualquier otro hecho de la vida del autor. El modo en que se organiza este conocimiento y se accede a él es muy crítico para el proceso de generación de historias.

La representación del conocimiento del autor está fuertemente influenciada por los conceptos propuestos por Schank en su modelo de la memoria dinámica (Schank, 1975; Schank y Abelson, 1975a). Según el autor, la memoria puede considerarse un esquema en el que las estructuras de conocimiento son revisadas cada vez que se incorpora nuevo conocimiento. En este esquema, la información relacionada con un tema determinado se estructura en subcadenas compartibles. Estas estructuras de memoria se denominaron MOPs (*Paquetes de Organización de la Memoria*), y pueden considerarse como fragmentos de conocimiento relacionados con un tema específico. En una representación basada en MOP, los fragmentos de conocimiento se interrelacionan entre sí y dan lugar a una estructura formada por heterarquías.

2.4.3. Representación basada en casos

Minstrel utilizaba un enfoque de razonamiento basado en casos para crear nuevas historias. También reutilizaba pasajes de historias generadas anteriormente, tomados de una memoria episódica (Malkewitz y Iurgel, 2006). Minstrel utilizaba una base de datos para apoyar este proceso. Esta base de datos contenía esquemas de historias que incluían los objetivos de los personajes, las acciones y los estados de los objetos, junto con el conocimiento de los objetivos del escritor dramático.

Los objetivos principales de Minstrel se dividían en cuatro categorías: tema de la historia, consistencia de la trama, narración dramática y presentación lingüística. Turner introdujo los Temas de Asesoramiento para la Planificación (PAT), que definió como temas que dan consejos sobre cómo debe actuar un planificador. Los PAT son objetivos más específicos diseñados para generalizar, especializar, mutar y recombinar los temas de la historia mientras el sistema está en marcha. El mundo ficticio de los personajes se modela mediante tres conceptos (Peinado y Gervás, 2006): Meta, Acto y Estado. Las metas y los actos se dividen en dos categorías: Character-Level y Author-Level, esta última orientada a la tarea de alto nivel relativa a la estructura narrativa del resultado. Los objetivos tienen números de prioridad (de 0 a 100) para que el planificador los ponga en orden. Los Métodos de Transformación-Recalificación-Adaptación (TRAM) se utilizan para mutar la historia con el fin de obtener resultados creativos.

Así, Minstrel ha creado una rica red de entidades vinculadas pertenecientes a diferentes dominios de conocimiento, gobernadas por una colección de operadores semánticos no tipificados y algoritmos de procesamiento definidos empíricamente. Además de esta estructura, hay *individuos fantasma* (Peinado y Gervás, 2006) que Minstrel utiliza para sus operaciones de planificación interna.

2.4.4. Representación basada en reglas

El conocimiento subyacente que impulsa la generación tiene un gran impacto en la calidad final del resultado en STellA. Puede decirse que se basa en gran medida en su conjunto de reglas centrales, utilizadas para representar el universo en el que se desarrollan sus historias.

Las reglas se consideran parte del dominio, y éstas operan básicamente produciendo secuencias de instantáneas y acciones (León y Gervás, 2014). Las instantáneas son estados del mundo. Una instantánea describe las posiciones exactas de los personajes, las afinidades, los objetos y cualquier otro detalle del mundo. Las acciones contienen información sobre lo que llevó al estado anterior al actual. Estas acciones se definen utilizando un vocabulario específico, con construcciones gramaticales concretas que definen una acción que realiza un actor en determinadas circunstancias o en un contexto determinado. Las instantáneas se definen de acuerdo con una ontología fija que estructura el mundo. En STellA, el mundo es una matriz, y cada entidad llena exactamente una celda. Las entidades grandes se componen de entidades pequeñas (ladrillos). Cada entidad tiene su propio conjunto de atributos.

2.4.5. Representación basada en ontologías

MakeBelieve (Liu y Singh, 2002) utilizaba un subconjunto de las ontologías contenidas en la base de representación del sentido común de Open Mind (Singh et al., 2002a) para describir la causalidad. Esta base de conocimiento contiene enunciados acerca del mundo. Algunos enunciados transmiten relaciones entre objetos o acontecimientos, expresadas como simples frases en lenguaje natural. Las relaciones causales binarias se extraían de estas frases y se almacenaban como trans-frames (Minsky, 1988). La base de conocimiento original de MakeBelieve ha sido continuada por la Open Mind Common Sense ConceptNet (Liu y Singh, 2004).

ConceptNet (Liu y Singh, 2004) era una base de conocimiento de sentido común abierta y un conjunto de herramientas de procesamiento del lenguaje natural que soportaba muchas tareas comunes de razonamiento textual sobre documentos del mundo real, incluyendo varios tipos de inferencias orientadas al contexto. Su base de conocimiento era una red semántica construida con un número considerable de afirmaciones relacionadas con el conocimiento del sentido común (más de un millón), que abarcaba los aspectos espaciales, físicos, sociales, temporales y psicológicos de la vida cotidiana. ConceptNet se generó automáticamente a partir de las frases del proyecto Open Mind Common Sense (Singh et al., 2002b), un proyecto de colaboración basado en la web.

Virtual Storyteller utilizaba ontologías para la creación de tramas (Oinonen et al., 2006). Partía de una ontología general básica que establecía una clasificación de objetos, y una ontología de acciones para definir un conjunto de acciones que los agentes de personajes podían llevar a cabo en el mundo de la historia, ambas representadas en el lenguaje de ontología web (OWL), estándar del W3C (Antoniou y Harmelen, 2004). Además, el Agente Mundial mantenía una descripción del mundo de la historia que era una instanciación de los objetos que existen en el entorno, sus propiedades y las relaciones entre ellos.

2.5. Generación de historias con múltiples tramas

Esta última sección del Estado de la Cuestión se centra en revisar los antecedentes en materia de generación automática de historias con múltiples tramas. Se puede decir que son pocos los casos de estudio previos, si bien existen sistemas que han ahondado en el tema y producido resultados destacables. Dada la importancia relativa de este mecanismo para gran parte de nuestra capacidad de comunicación, resulta sorprendente la escasa investigación centrada en la comprensión de los principios computacionales que lo rigen y que no exista un historial largo de generación y combinación de múltiples tramas en la generación automática de historias.

2.5.1. Sobre la trama en generación automática

Las similitudes entre una historia y un plan (ambos tienen un estado inicial y uno final, y un conjunto de pasos causalmente vinculados que conducen de uno a otro) ha llevado a la aparición de muchos esfuerzos para abordar la narración automática mediante soluciones de planificación de IA (Cohen y Feigenbaum, 2014). Dichas soluciones se basan en el concepto de un *operador de planificación*, una unidad de representación que expresa una acción, que declara explícitamente un conjunto de precondiciones de la acción a suceder y un conjunto de postcondiciones que se mantienen cuando ha sucedido. Los planificadores se basan en estas unidades para construir cadenas de acciones relacionadas causalmente desde un estado inicial hasta un objetivo. Un pionero en esta línea fue **TALE-SPIN** (Meehan, 1977), uno de los primeros generadores de historias, que escribía historias cortas sobre los habitantes de un bosque.

Aunque el concepto de trama no se mencionaba explícitamente en los artículos sobre TALE-SPIN, la naturaleza de su representación subyacente de las historias como planes da lugar a una estructura implícita de las historias, todas ellas basadas generalmente en la persecución de un único objetivo específico por parte del personaje protagonista. Por lo tanto, se limitaban a historias de una sola trama.

Una importante mejora en este campo llegó con **Autor** (Dehn, 1981b), un sistema de narración basado también en la planificación, pero que tenía en cuenta no sólo los objetivos de los personajes, sino un nivel adicional de objetivos del autor. Mientras que los objetivos de los personajes tienden a trazar una línea recta a lo largo de la historia, desde la situación inicial hasta el desenlace, los objetivos de los autores tienden a centrarse en la inserción de obstáculos en el camino del protagonista, para que la historia sea interesante para el lector.

A veces estos obstáculos adoptan la forma de personajes rivales con objetivos contrapuestos, lo que a veces puede llevar a la inserción de líneas argumentales adicionales. Estos sistemas iniciales intentaban generar historias cerradas, con un principio y un final claros. Un desarrollo diferente se introdujo en **Universe** (Lebowitz, 1984), un sistema que generaba guiones para una telenovela centrándose en las interacciones entre los personajes. Por su construcción, Universe funcionaba con una historia abierta, añadiendo episodios en cada tanda pero dejando siempre la historia abierta para ser continuada en episodios posteriores. Esto requería un enfoque de

generación basado en un amplio elenco de personajes y la introducción sistemática de complicaciones que afectaran a algunos de estos personajes para desencadenar los acontecimientos del nuevo episodio. Como resultado, las salidas generadas por Universe a lo largo de una secuencia de ejecuciones -un conjunto de episodios- pueden dar la impresión de una historia con múltiples tramas. Sin embargo, los procesos de generación empleados en cada momento no abordaban explícitamente la gestión de múltiples tramas, y no disponía de mecanismos específicos para manejar la intercalación de múltiples líneas argumentales. La consideración explícita de las intenciones de los personajes, para garantizar que las acciones de los personajes en las historias resultantes fueran creíbles, era el objetivo del **Fabulist** (Riedl y Young, 2010), que presentó toda una arquitectura para la generación y presentación automática de historias. Este sistema combina tanto los intereses del autor como la intencionalidad de los personajes. El hecho de que diferentes personajes tengan diferentes intenciones, y que el sistema las gestione todas con éxito, implica que muchos de los ingredientes para el manejo de historias con múltiples tramas pueden haber estado ya en este sistema, pero los aspectos específicos del entrelazamiento de tramas no se abordaron en los informes publicados del mismo.

La aparición de las gramáticas como herramientas de representación para captar la estructura de secuencias simbólicas complejas despertó el interés por el concepto de gramáticas de historias, (Rumelhart, 1975a; Thorndyke, 1977b). Varios narradores automáticos intentaron explotar este concepto. **GESTER** (*GEnerating STories from Epic Rules*) (Pemberton, 1989) era un sistema de generación de historias basado en reglas que gestionaba la información sobre la estructura de la historia en forma de una versión simplificada de una gramática narrativa que recogía los posibles acontecimientos y actores del subgénero épico.

Un ejemplo más elaborado fue **Brutus** (Bringsjord y Ferrucci, 1999), un sistema que generaba historias cortas sobre la traición. Utilizaba un modelo de conocimiento muy completo para representar el concepto y las implicaciones de la traición. También proporcionaba un modelo de generación basado en la gramática y un embellecedor literario, que le permitía generar historias de alta calidad, proporcionando textos que podrían haber sido escritos por humanos. En ambos sistemas, las gramáticas de los relatos constituyen la representación formal de la estructura de los relatos en la que se basan dichos sistemas. En este sentido, estas gramáticas captan de forma inherente la complejidad estructural del subconjunto del género que representan. Cuando este subconjunto contenga ejemplos de historias con múltiples tramas, la gramática captará la esencia de esta característica y permitirá la generación de historias de este tipo. Sin embargo, es probable que estas representaciones de ocurrencias de tramas múltiples en una gramática sean ejemplos de un posible entrelazamiento de tramas y no procedimientos aplicables en general a los problemas de entrelazamiento.

Un tercer enfoque es seguir modelos teóricos de cómo los humanos abordan la tarea. Este enfoque fue seguido por **Mexica** (Perez y Perez, 1999) un sistema de narración que generaba historias mitológicas sobre los mexicas, los primeros habitantes de México. Mexica se basaba en el modelo de compromiso y reflexión de la tarea de escritura (Sharples, 1999). Incluía un módulo de lectura capaz de procesar un corpus de historias anteriores del que obtenía las estructuras de conocimiento en

las que se basaba para generar nuevas historias. El funcionamiento del sistema es complejo, pero esencialmente se basaba en construir para cada historia previa una interpretación en términos de la evolución de las emociones de los personajes y de las tensiones entre ellos, y en abstraer de estas representaciones un conjunto de unidades de bloques de construcción de la trama que codificaban la probabilidad de que dos acontecimientos ocurrieran uno en el contexto del otro en una historia. MEXICA no tenía un concepto explícito de trama o línea argumental. La naturaleza de su funcionamiento permitía combinar en una historia cadenas de acontecimientos que implicaban a diferentes personajes, lo que daba lugar a la posibilidad de que aparecieran múltiples historias de línea argumental de un tipo simple entre sus resultados. Sin embargo, el sistema como tal no abordaba ninguno de los retos de manejar y entrelazar explícitamente múltiples tramas.

2.5.2. Sistemas generadores de historias con múltiples tramas

Uno de los trabajos destacables en este sentido es el algoritmo de tejido de tramas propuesto por Fay (2014). El sistema de Fay se basa en un corpus de historias escritas en inglés sencillo que puede leer y analizar en un conjunto de modelos de personajes y un conjunto de hilos narrativos para los distintos personajes. A partir de este material, se puede pedir al sistema que genere historias que combinen personajes de un tipo determinado. Para ello, busca los modelos de personajes que mejor se ajustan a los tipos dados, recupera los hilos narrativos asociados en el corpus con esos modelos y encuentra la mejor combinación de esos hilos narrativos en una sola historia. El sistema de Fay lee las historias utilizando la capacidad de comprensión de historias proporcionada por el sistema *Genesis*, desarrollado por el MIT (Winston, 2011, 2016).

Los hilos narrativos de los personajes elegidos se entretejen mediante un algoritmo que se asegura de que las tramas de los personajes sean compatibles y se encarga de construir una línea de tiempo coherente para todos los elementos argumentales de la historia. El procedimiento en sí es difícil desde el punto de vista computacional, ya que implica la selección del mejor conjunto de enlaces para los personajes principales de la historia que se solicitan a las entidades genéricas que aparecen como personajes secundarios en los hilos de otros personajes. A continuación, estos enlaces se utilizan para determinar el orden en el que los elementos de cada hilo se combinan en una única historia lineal.

Porteous et al. (2016) propusieron abordar el reto de construir historias con múltiples subtramas intercaladas, como las que aparecen regularmente en las series y telenovelas, en el contexto de un sistema de narración interactiva. Su sistema se basa en un enfoque basado en planes para la generación de narrativas con múltiples tramas. El sistema genera con éxito historias que se ajustan a diferentes parámetros de entrada, que especifican aspectos como el número de subtramas que se intercalan, y el tiempo relativo dedicado a la presentación de cada subtrama.

A diferencia de la solución de Fay, que operaba sobre un conjunto de tramas ya determinadas obtenidas de historias anteriores, esta solución construye las distintas subtramas que considera de forma incremental en el momento de entrelazarlas. En

este enfoque, se considera que una subtrama es una secuencia de *segmentos* de varios tipos centrados en un personaje concreto: normalmente un *introducción* seguido de un número variable de pares de *obstáculo* y *resolución*, con instancias de *exposición*, utilizadas para proporcionar información adicional al público en caso de necesidad, intercaladas opcionalmente entre ellas. Una historia con varias tramas es una secuencia de segmentos de este tipo en la que los segmentos adyacentes pertenecen a diferentes subtramas. Esto muestra cierta afinidad con la definición de trama de Fay, en la medida en que también se centra en un personaje concreto, pero incluye un número significativamente mayor de restricciones estructurales a nivel narrativo. El sistema funciona partiendo de un conjunto de objetivos para cada subtrama a considerar, y luego construyendo y presentando gradualmente segmentos para las subtramas, pasando de una a otra bajo la guía de los parámetros de entrada.

El sistema Raconteur (Chi y Lieberman, 2010, 2011) se centra en el problema, ligeramente diferente, de generar una historia que se ajuste a un conjunto determinado de hechos observados. Los hechos a considerar vienen como un conjunto de predicados de tiempo que describen los movimientos de las piezas en una partida de ajedrez. El sistema construye un conjunto de hilos narrativos, cada uno de los cuales agrupa los predicados que afectan a una pieza determinada (ya sea porque está involucrada en la acción o porque la acción tiene lugar dentro de su rango de percepción, es decir, dentro de las casillas del tablero suficientemente cercanas a ella). Las historias se construyen entonces mediante: la selección de un conjunto de piezas, la selección posterior de tramos concretos de sus hilos narrativos, y finalmente, la unión de estos tramos en una única secuencia lineal. El tejido se realiza teniendo en cuenta el orden cronológico relativo la proximidad física, la ocurrencia simultánea de piezas a través de hilos, y la naturaleza de las acciones capturadas en los predicados.

El sistema StoryFire de (Gervás, 2018) introduce un refinamiento en el proceso de tejido de hilos al considerar explícitamente el concepto de trama. Una trama en este contexto es una secuencia de elementos de la trama, cada uno de los cuales describe un evento relevante para la estructura de la historia, y un conjunto de roles que los personajes desempeñan en el evento. En una configuración similar a la del sistema Raconteur, los tramos de los hilos narrativos de determinadas piezas se seleccionan encontrando alineaciones optimizadas de partes del hilo con tramas de una base de datos. A continuación, el entrelazamiento de los hilos se basa en consideraciones tomadas de las tramas alineadas con ellos en los puntos potenciales para cambiar de hilo.

El sistema PlotAssembler desarrollado por Gervás (2019) aborda la exploración de un gran espacio de búsqueda de tramas aceptables mediante la combinación de pequeños tramos del discurso narrativo que no necesariamente aparecen de forma contigua en la secuencia de una historia. Un ejemplo de ello sería el secuestro de un personaje al principio de una historia que se relaciona con la liberación de ese personaje lograda hacia el final de la historia. Al utilizar un pequeño conjunto de unidades de este tipo, el proceso permite la construcción de muchas historias diferentes de complejidad arbitraria, siempre satisfaciendo las restricciones de coherencia y proporcionando un sentido de resolución. Por la naturaleza de estos segmentos -como puentes que se extienden desde un elemento en un punto de la historia hasta otros

no cercanos a ellos-, el procedimiento de construcción requerido para este proceso implica decisiones sobre cómo entrelazar estos segmentos. El sistema PlotAssembler aborda este problema mediante una solución basada en un corpus, en la que las probabilidades de que los elementos de diferentes segmentos aparezcan juntos en una historia vinculada por la presencia de determinados personajes se extraen de un corpus de historias anteriores.

2.6. Conclusiones

A lo largo de este capítulo se han revisado aspectos clave de las técnicas de generación automática de historias que se han ido desarrollando a partir de la historia de los propios sistemas generadores. Se ha puesto especial énfasis en la identificación de los principales problemas a los que se han enfrentado estos sistemas en su desarrollo, y que tienen también especial relevancia para la presente tesis.

En este sentido, cabe destacar la importancia de aspectos como la coherencia entre la trama y el contexto. La generación automática se enfrenta al desafío de crear historias con una trama coherente y bien estructurada, que tiene lugar en un determinado escenario o contexto. La creación de una trama implica una comprensión profunda del contexto de la historia, de los personajes, sus relaciones y sus emociones. Estos aspectos son especialmente fáciles de ver en sistemas como Mexica (Perez y Perez, 1999) o Scheherazade (Li et al., 2012, 2013).

Otro elemento clave es la creación de personajes interesantes y creíbles para el lector. Este es un desafío en la generación automática de historias. Los personajes deben ser coherentes con la trama y con la personalidad que se les ha asignado (Lebowitz, 1984; Faas, 2002; Swartjes, 2006; Si, 2010).

Si bien la representación de la causalidad, o las relaciones entre personajes, son relativamente tratables desde el punto de vista computacional, aspectos como la emotividad pueden resultar más difíciles de emular de forma efectiva. Las historias que funcionan son aquellas que logran conectar y crear emociones en el lector o el espectador. Es de destacar también el desafío de crear emociones que sean auténticas y creíbles como parte del proceso de generación. Algunos sistemas, como Mexica (Perez y Perez, 1999) o Brutus (Bringsjord y Ferrucci, 1999), se distinguen por sus aportaciones al respecto.

Finalmente, el aspecto básico que más impacto tiene en el presente trabajo es el problema de la consistencia, que es prácticamente una constante en los esfuerzos investigadores referidos. En los sistemas clásicos, se buscaban enfoques como la planificación basada en los objetivos de los personajes (Meehan, 1977; Lebowitz, 1984), o en los objetivos del autor de la historia (Dehn, 1981a), o en una combinación de enfoques (Yazdani, 1983). Los sistemas más recientes combinan varias estrategias y varios niveles de garantía de la consistencia, como por ejemplo, Slant (Montfort et al., 2013). Asegurar la consistencia interna de una historia es aún más difícil de resolver cuando se debe trabajar con varias tramas, que deben mantener una consistencia interna individualmente, y ser consistentes entre sí, según la forma en que se integren. En este sentido, son destacables los trabajos ya mencionados de Fay

(2014), centrado en explorar un universo de combinaciones de hilos de historias y personajes para identificar aquellas combinaciones más coherentes; o Porteous et al. (2016), basado en un enfoque distinto: construir historias con múltiples subtramas intercaladas de forma incremental.

Estos precedentes proporcionan un contexto para el análisis de los resultados, como se verá en el Capítulo 6, donde se analizan los problemas identificados y los resultados obtenidos en relación a la consistencia de las historias generadas en los diferentes sistemas desarrollados para la presente investigación.

Capítulo 3

Representación del conocimiento

"Conceptual integrity is the most important consideration in system design."
— Fred Brooks, The Mythical Man-Month

3.1. Resumen del modelo de representación del conocimiento

El presente capítulo presenta el modelo conceptual de representación del conocimiento desarrollado en la investigación. No existe una respuesta única al problema de cuál es el conocimiento mínimo necesario para crear historias, al igual que tampoco existe un sistema que emule completamente todas las capacidades humanas en la creación literaria. Cada sistema generador, o familia de sistemas generadores, ha concentrado sus esfuerzos en definir ese conjunto mínimo de datos necesarios, como se refleja en la sección 2.4 del Capítulo 2. El enfoque que se ha buscado en el presente trabajo ha sido proporcionar un mecanismo que maximice la compatibilidad entre los diferentes modelos de conocimiento, de forma que el modelo de representación se pueda extender fácilmente y soporte la mayor riqueza posible en la representación de historias. Esta flexibilidad en la representación está también relacionada con la modularidad del modelo arquitectónico, presentado en el Capítulo 4. Como se verá en dicho capítulo, cada componente de la arquitectura produce una parte de una historia y el soporte proporcionado por el modelo de representación es esencial para la construcción progresiva de la historia.

3.1.1. Elementos clave del modelo

Para la generación del modelo común de representación se ha llevado a cabo una exploración de los conceptos y entidades contemplados en los sistemas generadores existentes. Buena parte de este censo se encuentra recogido en el Capítulo 2.

El modelo de representación definido en la presente tesis se basa en una jerarquía de conceptos, tal como se muestra en la figura 3.1. Los conceptos han sido definidos

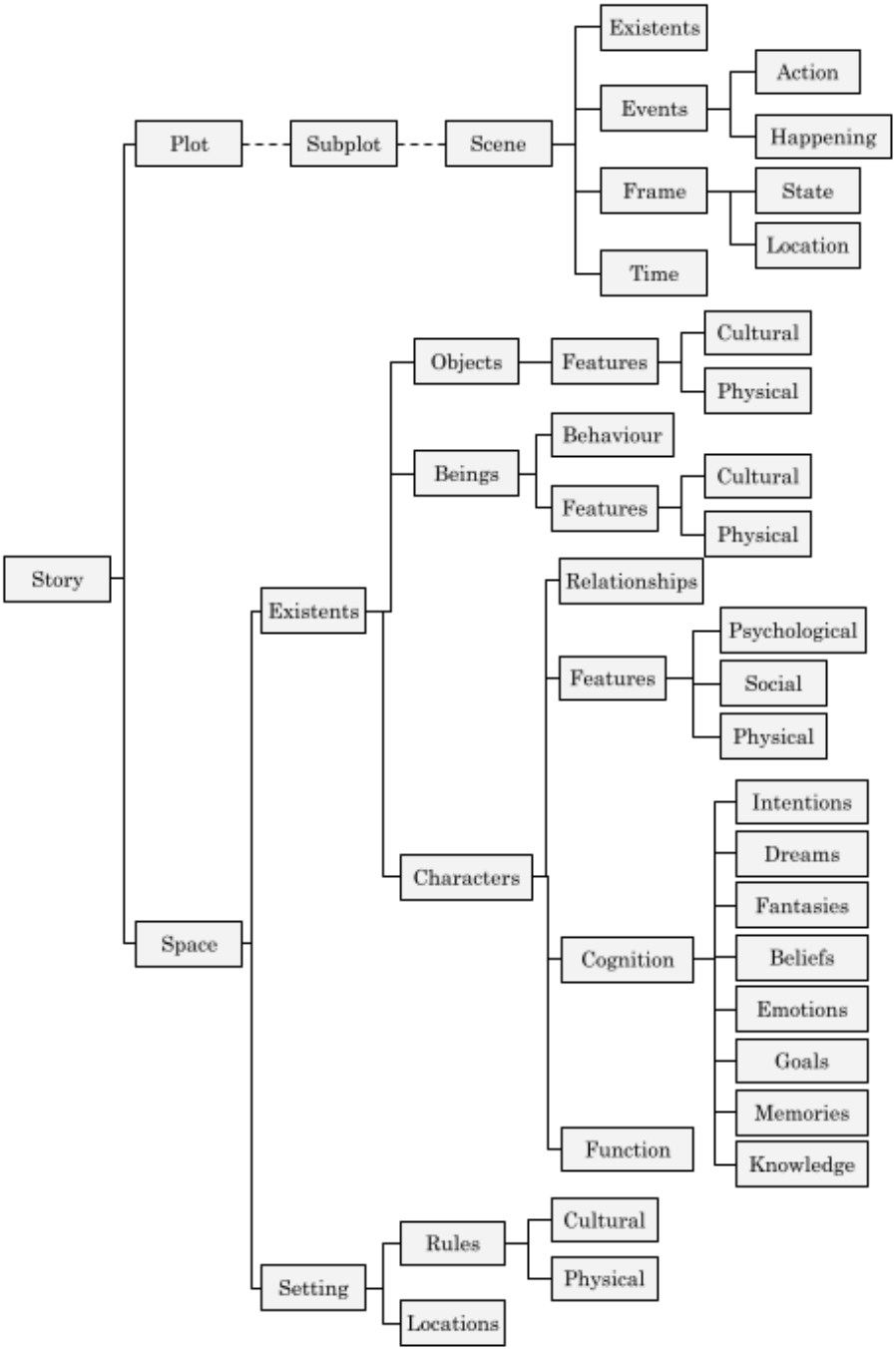


Figura 3.1: Modelo de representación del conocimiento de historias.

a partir de elementos de la Narratología clásica (Barthes, 1980; Chatman, 1980; Genette, 1980).

Una **historia** se compone de dos elementos esenciales: la trama, lo que sucede

en la historia siguiendo una línea temporal o causal; y el espacio, que representa al mundo (incluyendo a sus personajes) en el que se desarrolla la trama.

La **trama** se define en este modelo como una secuencia de escenas. Una **escena** está conceptualmente relacionada con la división de una obra de teatro, que representa un único episodio dentro de la trama. Está claramente condicionada por la división temporal, lo que significa que es una secuencia de acontecimientos que suceden durante un marco temporal. En el caso de las historias con múltiples tramas, lo que se define es una colección de **subtramas**, que a su vez son secuencias de escenas.

Desde el punto de vista espacial, cada escena está condicionada a tener lugar en un único **marco** espacial. Así, la escena está compuesta por una secuencia de **eventos**, que pueden ser acciones o sucesos. Una **acción** es un acto realizado por uno o varios personajes de la historia, que genera consecuencias. Las consecuencias resultantes de cada acción se expresan como una modificación en el estado global del espacio –considerándolo como el conjunto del escenario y los existentes–. Un **suceso** es un acontecimiento que ocurre en la trama, como accidente o como consecuencia de una acción o suceso anterior. Un suceso puede ser natural –llueve– o artificial –un accidente de coche.

El **espacio** abarca todo el universo en el que se desarrolla la trama. Está compuesto por el escenario y los existentes. Los existentes son el conjunto de actores que intervienen en la historia. Pueden ser personajes, seres vivos -un animal-, y un objeto del escenario. Los dos últimos tipos se definen principalmente por sus características físicas y su importancia cultural en la historia.

El **escenario** es una combinación de un conjunto de localizaciones físicas -o virtuales- en las que se desarrolla la acción de la historia, y el conjunto de reglas culturales y físicas que rigen el mundo de la historia. Las localizaciones pueden considerarse el escenario en el que se desarrolla cada escena que compone la trama. Así, como se muestra en el modelo, cada escena se vincula a su correspondiente localización.

Los **personajes** son habitualmente los elementos más relevantes en una historia, y también los más complejos de representar. El modelo tiene en cuenta no sólo sus características físicas, psicológicas y sociales, sino también las cognitivas. Otro elemento relevante de la representación de los personajes es la función. A pesar de que muchas aproximaciones a la representación de los personajes los tratan como entidades sujetas a reglas específicas que interactúan en un mundo narrativo simulado, existe otra línea de pensamiento importante en el tratamiento de los personajes: la visión funcional. En esta perspectiva, iniciada por Aristóteles (1974) y seguida por autores contemporáneos como Propp (1968), los personajes están subordinados a la acción narrativa. Hay modelos de conceptualización de los personajes (Propp, 1968; Greimas, 1983) que describen a los personajes en términos de una estructura profunda basada en sus funciones en la trama. Por lo tanto, la etiqueta de función se refiere a este enfoque y proporciona una forma de definir el papel funcional del personaje en la estructura subyacente de la historia.

La **cognición** de los personajes se representa de forma muy detallada debido a su importancia para garantizar la coherencia de la historia y la responsabilidad de los personajes. Los aspectos considerados han sido elegidos tras analizar los utilizados

por los sistemas de narración existentes (Si et al., 2006; León y Gervás, 2014; Dehn, 1981b; Lebowitz, 1985b; Méndez et al., 2016; Perez y Perez, 1999) y los estudios teóricos sobre Narrativa (Barthes, 1980; Margolin et al., 2013). Así, la representación de la cognición incluye, entre otras, facetas tales como los objetivos del personaje, las metas conscientes o inconscientes hacia las que se dirige el esfuerzo del personaje; las intenciones del personaje, el plan general que todo personaje tiene y que impulsa sus acciones; el conocimiento del personaje del mundo de le rodea; las emociones o sentimientos del personaje; o las creencias del personaje, axiomas verdaderos para el personaje al margen del mundo que le rodea. La cognición de los personajes se representa de forma muy detallada debido a su importancia para garantizar la coherencia de la historia y la responsabilidad de los personajes. Los aspectos considerados han sido elegidos tras analizar los utilizados por los sistemas de narración existentes y los estudios teóricos sobre Narrativa.

Desde el punto de vista de la estructura de la historia y la generación de la trama, es muy importante la representación de la **función** del personaje. Se trata de proporcionar una forma de representar los dos principales enfoques sobre el papel de los personajes en la trama. Hay modelos que consideran la trama como el resultado de las interacciones de los personajes en un mundo narrativo simulado, pero hay otra línea de pensamiento que considera que los personajes están subordinados a la acción narrativa. Hay sistemas de narración (Gervás, 2013) que describen a los personajes en términos de una estructura basada en sus funciones en la trama. Por lo tanto, la etiqueta de función se refiere a este enfoque y proporciona una manera de vincular el papel funcional del personaje a la estructura subyacente de la historia.

La **localización** es una combinación de un conjunto de lugares físicos -o virtuales- en los que se desarrolla la acción de la historia, y el conjunto de reglas culturales y físicas que rigen el mundo de la historia. La **localización** puede considerarse el escenario en el que se desarrolla cada escena que compone la trama. Así, como se muestra en el modelo, cada escena se vincula a su correspondiente localización.

El modelo de representación resultante se resume en la figura 3.1. Como se puede ver, la representación del conocimiento relativo a una historia abarca muchos aspectos. Esto se debe a la intención de mantener abiertas todas las posibles opciones que sean de interés para cualquier potencial componente nuevo que se integrase en la arquitectura.

3.1.2. Tratamiento del modelo en los artículos

Los artículos recogidos en el capítulo pueden agruparse en dos categorías principales: análisis de los elementos principales del conocimiento empleado por diferentes sistemas generadores y extracción de elementos comunes y particulares de cada uno; y elaboración de un modelo de representación maximalista a partir de dicho análisis. En la primera categoría entran los tres primeros artículos aquí recogidos, mientras que la segunda categoría se corresponde con el último artículo.

En el primer artículo, "Using CNL for Knowledge Elicitation and Exchange across Story Generation Systems" (Concepción et al., 2016b), se postula la utilización de un CNL, un lenguaje natural controlado, para extraer conocimiento de sistemas

generadores ya existentes y tratar de representarlo de acuerdo con un mecanismo (el CNL) que sea completamente abstracto a cualquier formalismo de representación empleado en dichos sistemas.

El artículo "Mining Knowledge in Storytelling Systems for Narrative Generation" (Concepción et al., 2016a) y el siguiente *challenge*, que puede verse como una continuación, "A Challenge Proposal for Narrative Generation using CNLs" (Concepción et al., 2016c), buscan extraer dicho conocimiento de sistemas concretos. En el primer caso, de STellA (León y Gervás, 2014), Charade (Méndez et al., 2016) y PropperWriter (Gervás, 2016); en el segundo caso, de cualquier sistema que deseara adherirse al *challenge*.

El último artículo, "A common model for representing stories in automatic storytelling" (Concepción et al., 2017b), expone una aproximación al modelo de generación del conocimiento casi completa, a falta del concepto de subtrama. El modelo de representación luego ha evolucionado hasta su forma definitiva, que es la descrita en la presente tesis.

3.2. Artículo: Using CNL for Knowledge Elicitation and Exchange across Story Generation Systems

Este artículo se enfoca en la utilización de CNL (*Controlled Natural Language*, Lenguaje Natural Controlado) para la extracción de conocimiendo de sistemas generadores de historias. De acuerdo con la definición de Fuchs y Schwitter (1995), un CNL "es un subconjunto del lenguaje natural que puede ser procesado con precisión y eficiencia por un computador, pero lo suficientemente expresivo como para permitir un uso natural por parte de no especialistas".

El artículo examina la aplicabilidad de un CNL para la representación del conocimiento en el contexto de la generación automática de historias. Y se centra en el estudio de un sistema generador concreto, STellA (León y Gervás, 2014).

Las conclusiones del artículo apuntan dos líneas principales: la primera, a la capacidad expresiva de un CNL para representar el conocimiento del dominio estudiado; la segunda, al fundamento que luego se desarrolla en artículos posteriores, la necesidad de que el modelo de representación recoja la **acción** que tiene lugar en la historia (y que más adelante se concretará en el concepto de *trama*) y la **caracterización del mundo** en el que sucede dicha acción (que se concretará en el concepto de *espacio*).

3.2.1. Cita completa

Concepción, E., Gervás, P., Méndez, G., León, C. (2016). *Using CNL for Knowledge Elicitation and Exchange Across Story Generation Systems.* In: Davis, B., Pace, G., Wyner, A. (eds) Controlled Natural Language. CNL 2016. Lecture Notes in Computer Science, vol 9767, pp. 81-91. Springer.

3.2.2. Resumen original

Story generation is a long standing goal of Artificial Intelligence. At first glance, there is a noticeable lack of homogeneity in the way in which existing story generation systems represent their knowledge, but there is a common need: their basic knowledge must be expressed unambiguously to avoid inconsistencies. A suitable solution could be the use of a controlled natural language (CNL), acting both as an intermediate step between human expertise and system knowledge and as a generic format in which to express knowledge for one system in a way that can be easily mined to obtain knowledge for another system – which might use a different formal language. This paper analyses the suitability of using CNLs for representing knowledge for story generation systems.

Using CNL for Knowledge Elicitation and Exchange Across Story Generation Systems

Eugenio Concepción[1], Pablo Gervás[2], Gonzalo Méndez[2( )], and Carlos León[1]

[1] Facultad de Informática, Universidad Complutense de Madrid, Madrid, Spain
{econcepc,cleon}@ucm.es
[2] Instituto de Tecnología del Conocimiento,
Universidad Complutense de Madrid, Madrid, Spain
{pgervas,gmendez}@ucm.es

Abstract. Story generation is a long standing goal of Artificial Intelligence. At first glance, there is a noticeable lack of homogeneity in the way in which existing story generation systems represent their knowledge, but there is a common need: their basic knowledge must be expressed unambiguously to avoid inconsistencies. A suitable solution could be the use of a controlled natural language (CNL), acting both as an intermediate step between human expertise and system knowledge and as a generic format in which to express knowledge for one system in a way that can be easily mined to obtain knowledge for another system – which might use a different formal language. This paper analyses the suitability of using CNLs for representing knowledge for story generation systems.

Keywords: CNL · Story generation · Knowledge representation

1 Introduction

Natural languages allow human communication and knowledge transmission, and they provide an unbeatable expressiveness for concept modelling and structuring. However, for the same reasons, they are substantially complex for automatic processing.

Controlled Natural Languages (CNLs) can be considered as a balance between the expressiveness of the natural languages and the need for a formal representation that can be handled by a computer. A CNL is an engineered subset of natural languages whose grammar and vocabulary have been restricted in a systematic way in order to reduce both ambiguity and complexity of full natural languages [1].

Against this background, CNLs are attractive because of two reasons: first of all, since they are subsets of natural languages, they are naturally easier to write and understand by humans than formal languages; secondly, they can be translated automatically (and often deterministically) into a formal target language and then be used for automated reasoning [1]. CNLs offer an additional advantage: unlike formal languages that require some degree of consensus concerning

B. Davis et al. (Eds.): CNL 2016, LNAI 9767, pp. 81–91, 2016.
DOI: 10.1007/978-3-319-41498-0_8

their syntax, a CNL should be more suitable for different teams to understand each other and therefore to more easily conclude an agreement. Of course, the application of formal languages such as XML, JSON or RDF formats is easy to achieve when considering interchange formats between systems that have established an agreed data model. This is not normally the case between systems that address storytelling from different perspectives, which are likely to have radically different data models. Interchange formats such as XML, JSON, or RDF will require a different translation procedure to convert knowledge built based on one data model to knowledge built on a different data model. Converting a given data model to and from a common CNL would allow every system to make their knowledge available to all other systems for which such a translation procedure is available.

Story generation systems are a form of expression for computational creativity. Using the words of Gervás [2], a story generator algorithm (SGA) refers to a computational procedure resulting in an artifact that can be considered a story. The term story generation system can be considered as a synonym of storytelling systems, that is, a computational system designed to tell stories.

2 Related Work

Storytelling systems require the representation and manipulation of large amounts of knowledge. This involves not only the product itself – stories represented at various levels of detail – but also the knowledge resources that are required to inform the construction processes. This section explores some of the aspects that need to be represented and some examples of how controlled natural language might be applied in specific cases.

The context in which the proposal presented in this paper occurs involves: the complexity of knowledge representation required for story generation systems, the already proven suitability of CNL in storytelling systems, the difficulties in eliciting the required knowledge, and existence of storytelling systems that already comtemplate automated transformations across different representation formats as an integral part of their functionality.

2.1 Knowledge Representation in Storytelling Systems

There are multiple dimensions when considering knowledge representation for story generation. Gervás and León [3] provided a list of the most relevant classifications, and proposed their own list of suitable dimensions obtained from the different aspects of a narrative: discourse, simulation, causality, character intention, theme, emotion, authorial intention, and narrative structure.

From a historical perspective, formal languages have been the most common way of knowledge representation. The reason for using formal languages is simplicity: they have a well-defined syntax, an unambiguous semantics and are very convenient for automated reasoning. Particularly, in the field of automatic story generation, there is an abundance of examples of this kind.

TALE-SPIN [4] is one of the earlier story generators that produced stories about the inhabitants of a forest. It was a planning solver system that took as inputs a collection of characters with their corresponding objectives, found a solution for characters goals, and finally wrote up a story narrating the steps performed for achieving those goals. TALE-SPIN knowledge representation relied on Conceptual Dependency Theory [5]. TALE-SPIN output can be defined as a trace through a problem-solving process where the problems were limited to a specific area of knowledge, named the problem domain, which was defined by a set of primitives, a set of goal states or problems, and procedures for achieving these goals. All this knowledge was expressed as a formal language.

Minstrel [6] was a story generation system that told stories about King Arthur and his Knights of the Round Table. Its building units were a collection of goals and the plans to satisfy them. Story construction in Minstrel operated as a two-stage process involving a planning stage and a problem-solving stage which reused knowledge from previous stories. The knowledge representation in Minstrel used an extension of a Lisp library called Rhapsody, a tools package that provided the user with ways to declare and manipulate simple frame-style representations, and a number of tools for building programs that used them. Minstrel used Rhapsody for defining frames, schemas with slots and facets which represent story themes or morals, dramatic effects, world states, characters beliefs and affects.

Mexica [7] was a computer model designed to generate short stories about the early inhabitants of Mexico. It used several knowledge structures for supporting its storytelling model: an actions library, a collection of stories for inspiring the new ones, and a group of characters and locations. The story generation process took as input a file of primitive actions for creating an in-memory data structure after processing. It also created additional structures by transforming the file of `Previous Stories` into the Concrete, `Abstract and Tensional Representations`. The data structure built by the initial step was called `Primitive Actions Structure`, and it served as a repository for the primitive actions, which consists of an action name and several sets representing characters and their circumstances. Relations in Mexica representing emotional links and tensions between characters were modelled by means of formal languages in terms of three attributes: type (love or friendship), valence (positive or negative) and intensity. Mexica knowledge base also contained stories created by humans representing well-formed narratives, expressed as action sequences.

MAKEBELIEVE [8] was a short fictional story generation system that used a subset of common sense from the ontology of the `Open Mind Common Sense Knowledge Base` [9] for describing causality. Binary causal relations were extracted from these sentences and stored as crude trans-frames. MAKEBELIEVEs original knowledge base has been continued subsequently by the `Open Mind Common Sense ConceptNet` [10]. A trans-frame [11] is a type of diagram used for representing the common information related to an action. Minsky used the Trans primitives from `Conceptual Dependency Theory` [5] as inspiration for trans-frame concept. Hence, these data structures can be used for representing a stereotyped situation.

2.2 Use of CNL in Storytelling Systems

There is not a long record of uses of CNL in the context of storytelling.

Inform [12] was a toolset for creating interactive fiction. From version 7 on, Inform provided a domain-specific language for defining the primary aspects of an interactive fiction like the world setting, the character features, and the story flow. The provided domain-specific language used a CNL, similar to Attempto Controlled English [13].

The StoryBricks [14] framework was an interactive story design system. It provided a graphical editing language based on Scratch [15] that allowed users to edit both the characters features and the logic that drove their behaviour in the game. By means of special components named story bricks, users could define the world in which characters live, define their emotions, and supply them with items. Story bricks were blocks containing words to create sentences in natural language when placed together. They served to define rules that apply under certain conditions during the development of the story in the game.

In the extended ATTAC-L version [16], authors introduced a model which combined the use of a graphical Domain Specific Modeling Language (DSML) for modelling serious games narrative, ATTAC-L [17], with a CNL to open the use of the DSML to a broader range of users, for which they selected Attempto Controlled English [13]. It allows describing things in logical terms, predicates, formulas, and quantification statements. All its sentences are built by means of two word classes: function words (determiners, quantifiers, negation words, etc.) and content words (nouns, verbs, adverbs and prepositions). The main advantage is that Attempto Controlled English defines a strict and finite set of unambiguous constructions and interpretation rules.

3 Knowledge Elicitation for Storytelling Systems

Storytelling systems are extremely knowledge hungry. Generated stories are only as good as the knowledge they have been derived from. Given the thirst for knowledge of story generation systems, knowledge elicitation has always been a significant concern for researchers in this area.

Recent attempts have been made to address this problem via crowdsourcing [18]. In this work, a number of human authored narratives are mined to construct a *plot graph*, which models the author-intended logical flow of events in the virtual world as a set of precedence constraints between plot events. Typical narratives in natural language on a given topic collected via Amazon Mechanical Turk (AMT). The crowd workers are required to follow a simplified grammar and a number of restrictions that resemble closely a CNL. These narratives are parsed and merged into a combined representation in terms of plot graph for the domain being explored, which is later used to inform the process of constructing narratives.

In this regard, the Genesis system [19] can also be considered as an interesting example of knowledge mining from stories written in simplified English. Genesis was developed for studying the story understanding process, including the human

ideological bias. It takes as inputs the text in Genesis English and a set of constraints representing the cultural and ideological context of the human reader that it will emulate. As a result, the system builds a graph-based representation using common sense rules. This knowledge structure allows the system not only to analyse problems, but also to answer questions and generate conclusions.

To inform the development of the Dramatis system for modelling suspense [20], O'Neill carried out an effort of knowledge engineering driven by methods adapted from qualitative research. The goal was to collect typical reader genre knowledge while simultaneously limiting engineer bias. The process was to acquire a corpus of natural language text and the conversion of that corpus into the knowledge structures required by Dramatis.

Although no CNL were strictly used for these tasks, the potential for their use in this context is clear.

3.1 Transformation Across Representations in Storytelling Systems

Gervás [21] attempts to model the procedures for composing narrative discourse from non-linear conceptual sources. It establishes algorithmic procedures for constructing a discourse – characterised by being a linear sequence of statements – to describe a set of facts known to have happened – which may involve events affecting a number of characters at different locations on overlapping periods of time. The composition procedure starts from a description of the set of events to be considered, produces an intermediate representation that captures the typical human view of events – restricted to what might have been perceived by a given character at a given moment in time – and proposes an algorithmic procedure to build a sequence of spans of narrative discourse, each capturing perception by an individual character. These spans of narrative discourse are rendered first as a simple conceptual description and then as pseudo-text. Table 1 shows examples of the original input as algebraic notation for a chess game, the conceptual representation of the composed discourse, and the pseudo-text rendering.

This conceptual description is built in such a way that the system can interpret it to reconstruct a version of the exhaustive description of the world that it started from. This is used by the system to validate the decisions taken during the composition of the discourse. The ability to convert automatically from one to another across different formats of knowledge representation can play a crucial role in the proposal described in this paper.

4 Using CNL for Knowledge Elicitation and Exchange Across Story Generation Systems

There are two important conclusions that can be extracted from the material presented so far.

First, that story generations are faced with a significant challenge of acquiring knowledge resources in the particular representation formats that they use. The difficulty of expressing human knowledge in formal languages is a considerable

Table 1. Original input as algebraic notation for a chess game, conceptual representation of the composed discourse, and pseudo-text rendering.

		The fifth black pawn was three squares north of the centre of the board. The fifth black pawn found the black left bishop was behind. The fifth black pawn found the fourth black pawn was same. The fifth black pawn found the black queen was behind. The fifth black pawn found the black king was behind. The fifth black pawn saw the fourth black pawn moving south a square. Four days later, the fifth black pawn saw the black right knight arriving left. ...
	focalizes_on bp5	
1. e4 e5	9. Nc3 Qe8	is_at lbb behind
2. Nf3 Nc6	10. Bxf6 gxf6	is_at bp4 same
3. Bc4 Bc5	11. Nh4 Kh8	is_at bp4 same
4. O-O Nf6	12. Qh5 Be6	is_at bq behind
5. h3 d6	13. Qxh6+ Kg8	is_at bk behind
6. d3 O-O	14. Nd5 Bxf2+	move_to bp4 1/south
7. Bg5 h6	15. Rxf2 Qd7	4 later
8. Bh4 b6	16. Nxf6# 1-0	arrives_at rbk left
	...	

obstacle. In this context, the use of a CNL would provide the means for quicker development of required resources in a format easier to write for human experts. There are ongoing efforts to build such information via crowdsourcing and/or to learn this information via information extraction techniques, but all efforts along these lines either:

– have met with limited success,
– need to rely on huge amounts of hand annotation,
– require procedures of controlled edition very similar to CNL.

Second, that every story generation system defines its own format for knowledge representation, optimised to support its storytelling process. Although the development of a common formalism for representing knowledge would provide a major breakthrough, this is unlikely to happen in the near future (see Gervás and León [3] for a detailed discusion of the problems involved). Under the circumstances, the use of a CNL for codifying resources for storytelling systems might provide some relief. If authors of storytelling systems were to develop the initial version of their resources in a commonly agreed CNL, and then develop the appropriate automated transformations to generate knowledge in their own preferred format, the same resources written in CNL might be of use to researchers developing different storytelling systems. By simply writing the appropriate transformations into their own preferred format, much of the already available knowledge could be reused.

The main advantage of using a CNL is that it can be expressed by domain experts and then it can be translated to the variety of formal languages used

in different systems. This feature allows the creation of a common language not only for expressing the different aspects involved in narrative generation, but also for exchanging knowledge resources across different storytelling systems. This might also pave the way for the development of common benchmarks for testing storytelling systems. A relevant conclusion mentioned by Gervás and León [3] is that the same information may be represented through different data structures without affecting its essence, or a data structure can be extended for representing additional types of information.

5 A Case Study: The STellA System

STellA (Story Telling Algorithm) [22,23] is a story generation system that controls and chooses states in a non-deterministically generated space of partial stories until it finds a satisfactory simulation of events that is rendered as a story. This simulation has been modeled as a knowledge intensive approach in which the whole world domain is explicitly represented as a simplistic view of a realistic environment. At each step, candidate updated versions of the current state are computed and the most likely ones are identified by computing their likelihood in terms of their plausibility and their narrative properties. Candidate partial stories are evaluated based on how well they satisfy a given set of constraints and how their tension curves compare with a set of target curves. The results of this process are used to decide when a partial story is promising and whether a story is finished.

5.1 Knowledge Representation in STellA

STellA is a resource hungry system. The underlying knowledge driving the generation has a big impact on the final quality of the output. One of the main characteristics of STellA is its heavy dependency on a core set of rules defining the whole universe the system is capable of producing stories about. The application of a CNL can highly improve the situation by letting external sources be added as knowledge.

The rules are considered part of the domain, and these basically operate by producing sequences of *snapshots* and *actions*. Snapshots are states of the world. A snapshot describes exact character positions, affinities, items, and every other detail of the world. Actions contain information about what led from the previous state to the current one. Then, a story is formally defined as show in Eq. 1.

$$story = \{(s_1, [a_{1,1}, a_{1,2}, \ldots, a_{1,n}]), \ldots, (s_z, [a_{z,1}, a_{z,2}, \ldots, a_{z,n})\} \tag{1}$$

where s_x is a snapshot and $a_{i,j}$ is an action. Each pair is called a *state*, and a sequence of states form a story. Actions have their own vocabulary and correspond to specific structures like $take(character, item)$ or $approach(character, place)$. Snapshots are defined according to a fixed ontology that structures the world. In STellA, the world is a matrix, and every entity fills exactly one cell. Big entities, as houses, are composed of small entities (bricks).

Each entity has its own set of attributes. Characters, in particular, are the most developed and detailed entities and are described in terms of properties commonly influencing narratives:

- physical properties for moving and interacting with the environment,
- affinities between characters,
- an internal representation of the world, which does not have to be the same as the real world. Characters use this for planning.
- Roles (moral tendency) and traits (special capabilities)

Rules, then, receive a current state (a pair of a snapshot and the actions that led to it from the previous one) and non-deterministically output a new set of actions. The non-deterministic aspect is not relevant for this discussion, and more information can be found on the literature [23]. The creation of rules is, then, driven by the data model.

5.2 Fundamental Features of a CNL for STellA

In order to design a CNL for the knowledge base of STellA, the data model must be covered by the language. While it is not a trivial task, the fact that the data model is well established and structured means the vocabulary can be easily described and expressed in natural language. In particular, the sentences to use must be able to describe preconditions and actions. Let us examine the following case:

> When a character is alive and it moves north, its y coordinate has to be increased.

State information and actions are used in the sentence:

- *a character is alive* refers to a state. The rule is valid for *any* character that is alive.
- *it moves north* is an action, it corresponds to $move(?x, north)$.
- *its* y *coordinate must be increased* applies a change in the state: $y = y + 1$.

In order to transform the sentence into a rule, a template must be filled in like this:

precondition $\forall\ ?c\ \in\ Characters,\ alive(?c)$
action $move(?c, north)$
postcondition $?c.y\ \leftarrow\ ?c.y + 1$

Quantified state and action information must be addressed, which is, given the data model, doable. The problem is the generality needed in the kind of changes that must be applied in the postconditions. In this case, the simple *move* rule has updated the value of the y location component of the character $?c$. For more complex changes (like, for instance, traversing all available items in $?c$'s inventory), the CNL should cover a non-trivial set of constructions.

It is hypothesized that these complex constructions are, in general, mostly covered by a number of basic operations (traversing lists, accessing elements with given properties or finding elements in the world). Making this constructions atomic and suitable for composition can led to a simpler, reasonable expressive CNL. More in-depth study must be conducted in order to gain better insight on what this set of properties is.

6 Discussion

The construction of a CNL with the desired properties is a significant challenge. Namely because, to cover all the various aspects of representation relevant for storytelling systems as a whole, it would have to address at least all aspects described by Gervás and León [3]. From a simplified point of view, two major layers of representation can be considered. Firstly, an orchestration layer, whose main concern is related to the dynamic flow of the story. Secondly, a characterization layer, which is focused on representing the static features of the elements that define the story. The orchestration layer is related to the discourse sequence aspect, the causal aspect, and the intentional aspect. On the other side, the characterization layer includes the remaining aspects: the simulation aspect, the theme aspect, the emotional aspect, the authorial aspect, and the narrative structure aspect. It is both necessary and important to emphasise that these layers, and the aspects associated to them, are mutually interwoven. This means that changes in the data related to one aspect typically will cause changes in other aspects. For example, a change in the feelings of a character (emotional aspect) could determine his/her course of action, or modify his/her objectives (intentional aspect).

In addition, the use of a domain-specific glossary would serve not only for establishing a proper definition of the knowledge domain, but also for reducing the risk of polysemy. One of the potential issues with CNL is that they are not specifically designed to address word sense disambiguation. The definition of a CNL usually focuses on analysing just some key words that are relevant for building the discourse representation structure.

In a first-approach, it is possible to define a basic modelling for just some aspects of the narration. One suitable candidate could be the sequential aspect through the use of a planning modelling language, since a good part of the story generation systems architecture is based on a planner. A common language for modelling planners is PDDL (Planning Domain Description Language) [24], which is designed to formalize dynamic models, where actions guide the model through a series of state. This first step would consist of developing a match between the knowledge modelled in PDDL and a collection of primitives for describing the same information.

7 Conclusions and Future Work

This paper discussed the suitability of using a CNL for eliciting and exchanging knowledge in the context of a range of story generation systems. As shown above,

there have been precedents of the use of CNL in the interactive storytelling domain with satisfactory results. The advantages of using a CNL for elicitation of knowledge resources had been demonstrated in the past. The potential for providing a compatible format for the exchange of knowledge across systems would be a major positive contribution to the field.

Future work involves not only the complete development of a CNL for covering the knowledge needed in this domain, but also the development of evaluation techniques that validate the suitability and portability of this representation over a wide range of story generation systems. In particular, a short term goal of the authors is to establish a CNL that can serve to develop knowledge resources that might as common ground data for a shared evaluation task for storytelling systems.

Acknowledgements. This paper has been partially supported by the project WHIM (611560) funded by the European Commission, Framework Program 7, the ICT theme, and the Future Emerging Technologies FET program.

References

1. Schwitter, R.: Controlled natural languages for knowledge representation. In: Proceedings of the 23rd International Conference on Computational Linguistics: Posters. COLING 2010, pp. 1113–1121. Association for Computational Linguistics, Stroudsburg (2010)
2. Gervás, P.: Story generator algorithms. In: The Living Handbook of Narratology. Hamburg University Press (2012)
3. Gervás, P., León, C.: The need for multi-aspectual representation of narratives in modelling their creative process. In: 5th Workshop on Computational Models of Narrative. OASIcs-OpenAccess Series in Informatics (2014)
4. Meehan, J.R.: Tale-spin, an interactive program that writes stories. In: Proceedings of the Fifth International Joint Conference on Artificial Intelligence, pp. 91–98 (1977)
5. Schank, R.C., Abelson, R.P.: Scripts, plans, and knowledge. In: Proceedings of the 4th International Joint Conference on Artificial Intelligence, vol. 1, IJCAI 1975, pp. 151–157. Morgan Kaufmann Publishers Inc., San Francisco (1975)
6. Turner, S.R.: Minstrel: A Computer Model of Creativity and Storytelling. Ph.D. thesis, University of California at Los Angeles, Los Angeles, CA, USA (1993). UMI Order no. GAX93-19933
7. Perez y Perez, R.: MEXICA: A Computer Model of Creativity in Writing. Ph.D. thesis, The University of Sussex (1999)
8. Liu, H., Singh, P.: Makebelieve: using commonsense knowledge to generate stories. In: Dechter, R., Sutton, R.S. (eds.) AAAI/IAAI, pp. 957–958. AAAI Press / The MIT Press (2002)
9. Singh, P., Lin, T., Mueller, E.T., Lim, G., Perkins, T., Zhu, W.L.: Open mind common sense: knowledge acquisition from the general public. In: Meersman, R., Tari, Z. (eds.) CoopIS 2002, DOA 2002, and ODBASE 2002. LNCS, vol. 2519, pp. 1223–1237. Springer, Heidelberg (2002)
10. Liu, H., Singh, P.: ConceptNet a practical commonsense reasoning tool-kit. BT Technol. J. **22**(4), 211–226 (2004)

11. Minsky, M.: Society of mind. Simon and Schuster, New York (1988)
12. Reed, A.: Creating Interactive Fiction with Inform. Cengage Learning, Boston (2010)
13. Fuchs, N.E., Schwertel, U., Schwitter, R.: Attempto controlled english – not just another logic specification language. In: Flener, P. (ed.) LOPSTR 1998. LNCS, vol. 1559, pp. 1–20. Springer, Heidelberg (1999)
14. Campbell, M.: A new way to play: make your own games. New Sci. **211**(2829) (2011)
15. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., Kafai, Y.: Scratch: programming for all. Commun. ACM **52**(11), 60–67 (2009)
16. Van Broeckhoven, F., Vlieghe, J., De Troyer, O.: Using a controlled natural language for specifying the narratives of serious games. In: Schoenau-Fog, H., et al. (eds.) ICIDS 2015. LNCS, vol. 9445, pp. 142–153. Springer, Heidelberg (2015). doi:10.1007/978-3-319-27036-4_13
17. Broeckhoven, F.V., Troyer, O.D.: Attac-l: A modeling language for educational virtual scenarios in the context of preventing cyber bullying. In: 2nd International Conference on Serious Games and Applications for Health, pp. 1–8. IEEE, May 2013
18. Li, B., Lee-Urban, S., Johnston, G., Riedl, M.O.: Story generation with crowdsourced plot graphs. In: Proceedings of the 27th AAAI Conference on Artificial Intelligence, AAAI 2013 (2013)
19. Winston, P.H.: The genesis story understanding and story telling system a 21st century step toward artificial intelligence. Technical report, Center for Brains, Minds and Machines (CBMM) (2016)
20. O'Neill, B.: A Computational Model of Suspense for the Augmentation of Intelligent Story Generation. Ph.D. thesis, Georgia Institute of Technology, Atlanta, Georgia (2013)
21. Gervás, P.: Composing narrative discourse for stories of many characters: a case study over a chess game. Literary Linguist. Comput. **29**(4), 511–531 (2014)
22. León, C., Gervás, P.: A top-down design methodology based on causality and chronology for developing assisted story generation systems. In: 8th ACM Conference on Creativity and Cognition, Atlanta, November 2012 (2011)
23. León, C., Gervás, P.: Creativity in story generation from the ground up: non-deterministic Simulation driven by Narrative. In: 5th International Conference on Computational Creativity, ICCC 2014, Ljubljana, Slovenia (2014)
24. McDermott, D., Ghallab, M., Howe, A., Knoblock, C., Ram, A., Veloso, M., Wilkins, D.: PDDL - the planning domain definition language (1998)

3.3. Artículo: Mining Knowledge in Storytelling Systems for Narrative Generation

Este artículo desarrolla los elementos presentados en el anterior (Concepción et al., 2016b) para extenderlos en dos direcciones: por un lado, se presenta una primera solución para la extracción de conocimiento y su volcado a una representación en CNL de tres sistemas generadores: STellA (León y Gervás, 2014), Charade (Méndez et al., 2016) y PropperWriter (Gervás, 2016); por otro lado, se establecen las dimensiones que deberá contemplar la representación del conocimiento a partir de las propuestas por Gervás y León (2014).

Los resultados presentados en este artículo en relación a las dimensiones de la representación: simulación, causalidad, intencionalidad, temática, emotividad, estructura narrativa, secuencia del discurso y enfoque del autor; tendrán una fuerte influencia en la forma de estructurar la representación presentada en el siguiente artículo (Concepción et al., 2017b).

3.3.1. Cita completa

E. Concepción, P. Gervás, and G. Méndez (2016). *Mining Knowledge in Storytelling Systems for Narrative Generation.* In Proc. of the INLG 2016 Workshop on Computational Creativity in Natural Language Generation, pages 41–50, Edinburgh, UK. Association for Computational Linguistics.

3.3.2. Resumen original

Storytelling systems are computational systems designed to tell stories. Every story generation system defines its specific knowledge representation for supporting the storytelling process. Thus, there is a shared need amongst all the systems: the knowledge must be expressed unambiguously to avoid inconsistencies. However, when trying to make a comparative assessment between the storytelling systems, there is not a common way for expressing this knowledge. That is when a form of expression that covers the different aspects of the knowledge representations becomes necessary. A suitable solution is the use of a Controlled Natural Language (CNL) which is a good half-way point between natural and formal languages. A CNL can be used as a common medium of expression for this heterogeneous set of systems. This paper proposes the use of Controlled Natural Language for expressing every storytelling system knowledge as a collection of natural language sentences. In this respect, an initial grammar for a CNL is proposed, focusing on certain aspects of this knowledge.

Mining Knowledge in Storytelling Systems for Narrative Generation

Eugenio Concepción and **Pablo Gervás** and **Gonzalo Méndez**
Facultad de Informática
Instituto de Tecnología del Conocimiento
Universidad Complutense de Madrid
{econcepc,pgervas,gmendez}@ucm.es

Abstract

Storytelling systems are computational systems designed to tell stories. Every story generation system defines its specific knowledge representation for supporting the storytelling process. Thus, there is a shared need amongst all the systems: the knowledge must be expressed unambiguously to avoid inconsistencies. However, when trying to make a comparative assessment between the storytelling systems, there is not a common way for expressing this knowledge. That is when a form of expression that covers the different aspects of the knowledge representations becomes necessary. A suitable solution is the use of a Controlled Natural Language (CNL) which is a good half-way point between natural and formal languages. A CNL can be used as a common medium of expression for this heterogeneous set of systems. This paper proposes the use of Controlled Natural Language for expressing every storytelling system knowledge as a collection of natural language sentences. In this respect, an initial grammar for a CNL is proposed, focusing on certain aspects of this knowledge.

1 Introduction

Natural language is the most basic form of knowledge representation for the humans, because it allows communication and knowledge transmission. Natural languages provide an unbeatable expressivity for concept modelling and structuring. However, for the same reasons they are substantially complex for automatic processing.

A Controlled Natural Language (CNL) is an engineered subset of natural languages whose grammar and vocabulary have been restricted in a systematic way in order to reduce both ambiguity and complexity of full natural languages (Schwitter, 2010). CNL can be considered as a tradeoff between the expressivity of the natural languages, and the need for the orthogonality of a formal representation that can be handled by a computer.

Story generation systems are a form of expression for computational creativity. According to (Gervás, 2012), a story generator algorithm (SGA) refers to a computational procedure resulting in an artefact that can be considered a story. The term story generation system can be considered as a synonym of storytelling systems, that is, a computational system designed to tell stories.

Story generation system are faced with a significant challenge of acquiring knowledge resources in the particular representation formats that they use. They face a inherent difficulty when using formal languages in the detachment between the formulation of the needs in the real world and its representation in a formal construction.

In this context, the use of a CNL would provide the means for a quicker development of required resources in a format easier to write for human experts. So, the use of a CNL for codifying resources for storytelling systems might provide some advantage. If authors of storytelling systems were to develop the initial version of their resources in a commonly agreed CNL, and then develop the appropriate automated transformations to generate knowledge in their own preferred format, the same resources writ-

Proceedings of the INLG 2016 Workshop on Computational Creativity and Natural Language Generation, pages 41–50,
Edinburgh, September 2016.

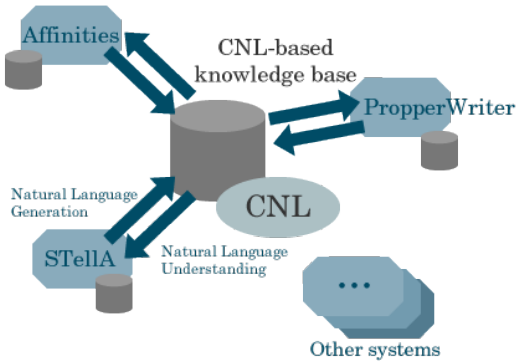


Figure 1: Architecture of the shared CNL-based knowledge base system.

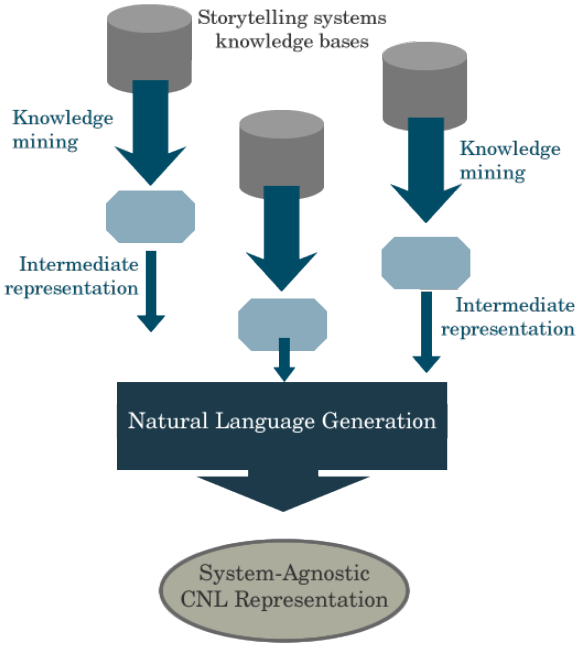


Figure 2: Knowledge Mining process applied to Natural Language Generation.

ten in CNL might be of use to researchers developing different storytelling systems. Some previous studies on this matter can be found in (Schwitter, 2010), (Kuhn, 2009), (Power et al., 2009), (Davis et al., 2009), (Fuchs et al., 2008), and (Funk et al., 2007).

Particularly, the use of a CNL for knowledge representation has been documented previously (Kuhn, 2009), and (Barzdins, 2014). In both cases, these precedent works are quite convenient with respect to information extraction of and reasoning with the content of texts.

This paper proposes a model of a Controlled Natural Language understood as a means for mining knowledge from existing storytelling systems.

This process is part of a wider project which aims at the development of a collaborative environment involving several story generation systems.

The purpose of this environment is to establish a co-creation architectural model which allows the involved systems to take advantage of a shared knowledge base and use it for enhancing the quality of the generated texts. The architecture of this system is schematically depicted in the Figure 1. In the current stage of the model, the NLG step is used for translating the system-specific formalisms into the common CNL statements. In the final model, there will be an additional NLG step when generating the refined story.

The thrust of this approach is the use of a CNL as a shared representation for the various knowledge models of the different story generation systems. Ideally, every custom representation should be translated into the common CNL for being subsequently employed. In this way, the story generation process will result in a build-up of contributions from different systems. For example, one of the depicted systems, labeled as *Affinities* (Méndez et al., 2016), is specialized in characters interactions and affinities, but it lacks in a deeper enhancement of the narrative discourse, that can be compensated by STellA (León and Gervás, 2011).

The drafted architecture aims at several objectives. Firstly, it intends to establish a collaborative model that allows the free exchange of knowledge between the different storytelling systems in order to develop an iterative improvement process of literary creation. Beside this objective, it promotes the development of a knowledge representation model for creating a common, system-agnostic knowledge base that can be feed in the future with the outcomes of new storytelling systems, without the need to adapt their knowledge representation models.

The scope of this paper relates only to the Natural Language Generation step, as depicted in the Figure 2.

2 Related Work

2.1 Natural Language Generation

The process of Natural Language Generation has been clearly defined (Reiter et al., 2000), as well as its six distinctive tasks:

- Content Determination: in which the generation system makes a decision concerning the information that will be taken into account for generating the text.
- Discourse Planning: this task involves decisions about how the text should be globally articulated.
- Lexicalisation: in which the generation system makes a choice of the particular words and phrases it considers suitable to convey the semantics of the selected information, in the given natural language and its context.
- Aggregation: this task involves decisions concerning the composition of the generated sentences to form a natural discourse.
- Referring Expression Generation: this task involves the determination of the properties of a given linguistic element, which to be used when the element is mentioned again.
- Surface Realisation: the last task that reviews the text for checking that it presents syntactically, morphologically and orthographically correct sentences in the corresponding natural language.

Many different architectures have been proposed for NLG systems, reflecting the range of different applications and its purposes. Basically, there are two main models of the NLG process: the Abstract Generation System (Bateman, 1997) and the Abstract Referential Model, an outcome of the Reference Architecture for Generation Systems (RAGS) (Cahill et al., 2000) which is aiming at standards for NLG architecture.

2.2 Knowledge Representation in Storytelling Systems

From an historical perspective, formal languages have been the most common way of knowledge representation. The reason for using formal languages is simplicity; they have a well-defined syntax, an unambiguous semantics and they are very convenient for automated reasoning. Particularly, in the field of automatic story generation, there is an abundance of examples of this kind.

TALE-SPIN (Meehan, 1977a) was one of the earlier story generators. It was a problem solver, top-down and goal-directed story generation engine. TALE-SPIN generated stories about the inhabitants of a forest taking a collection of characters with their corresponding objectives as inputs. TALE-SPIN found a solution for those characters goals, and wrote up a story narrating the steps performed for achieving those goals.

TALE-SPIN knowledge representation relied on Conceptual Dependency Theory (Schank and Abelson, 1975). It used a set of primitives for representing the problem domain. All its knowledge was expressed as a formal language.

Minstrel (Turner, 1993) was a story generation system that told stories about King Arthur and his Knights of the Round Table. Each story was focused on a moral, which also provided the seed for developing the story.

The knowledge representation in Minstrel used an extension of a Lisp library called Rhapsody. Rhapsody was a tools package for AI program development that provided the user with ways to declare and manipulate simple frame-style representations, and a number of tools for building programs that use them.

Mexica (Perez y Perez, 1999) was developed as a computer model whose purpose was studying the creative process. It generated short stories about the early inhabitants of Mexico. Mexica was a pioneer in that it took into account emotional links and tensions between the characters as a means for driving and evaluating ongoing stories.

Mexica used several knowledge structures for supporting its storytelling model: An actions library, a collection of stories for inspiring the new ones, and a group of characters and locations. The generation process also took several steps, in which data were progressively transformed.

Mexica knowledge management involves several concerns in order to provide a high-quality outcome, in terms of literary production. Its knowledge base included several types of structures for representing

things like characters relationships, actions, emotional links, and a literary base composed of previously generated stories.

Brutus (Bringsjord and Ferrucci, 2000) was a system that generated short stories using betrayal as leitmotiv. It had a rich logical model for representing betrayal. This feature allowed it to generate complex stories. A very innovative aspect of Brutus was that it considered the existing body of knowledge about literature and grammar for generating stories.

Brutus structured its knowledge in several layers, including a grammar specific part. So, the process of converting the plot into the final output was carried out by the application of a hierarchy of grammars: story grammars, paragraph grammars and sentence grammars. This hierarchical procedure led to define every story as a sequence of paragraphs which in turn were sequences of sentences.

MAKEBELIEVE (Liu and Singh, 2002) was a short fictional story generation system that used common sense knowledge to generate stories. The user provided a story about a character as initial seed, and then MAKEBELIEVE attempted to continue that story by inferring possible sequences of events that might happen to that character. The system used common sense knowledge about causality and how the world works, mined from the Open Mind Common Sense knowledge base (Singh et al., 2002).

STellA (Story Telling Algorithm) (León and Gervás, 2014) is a story generation system that controls and chooses states in a non-deterministically generated space of partial stories until it finds a satisfactory simulation of events that is rendered as a story.

STellA uses a custom representation for the knowledge it needs. It manages several different structures, including a matrix representation of the world in which characters live, and a set of rules for evaluating the range of results associated to the actions.

2.3 Use of CNLs in Storytelling Systems

There is not a long record of application of CNLs in the context of storytelling.

Inform (Reed, 2010) was a toolset for creating interactive fiction. As from version 7, Inform provided a domain-specific language for defining the primary aspects of an interactive fiction like the world setting, the character features, and the story flow. The provided domain-specific language used a CNL, similar to Attempto Controlled English (Fuchs et al., 1998).

The StoryBricks (Campbell, 2011) framework was an interactive story design system. It provided a graphical editing language based on Scratch (Resnick et al., 2009) that allowed users to edit both the characters features and the logic that drove their behaviour in the game. By means of special components named story bricks, users could define the world in which characters live, define their emotions, and supply them with items. Story bricks were blocks containing words to create sentences in natural language when placed together. They served to define rules that apply under certain conditions during the development of the story in the game.

In the extended ATTAC-L version (Broeckhoven et al., 2015), authors introduced a model which combined the use of a graphical Domain Specific Modeling Language (DSML) for modelling serious games narrative, ATTAC-L (Broeckhoven and Troyer, 2013), with a CNL to open the use of the DSML to a broader range of users, for which they selected Attempto Controlled English (Fuchs et al., 1998). It allows describing things in logical terms, predicates, formulas, and quantification statements. All its sentences are built by means of two word classes: function words (determiners, quantifiers, negation words, etc.) and content words (nouns, verbs, adverbs and prepositions). The main advantage is that Attempto Controlled English defines a strict and finite set of unambiguous constructions and interpretation rules.

3 Conceptual Basis

Towards the definition of a shared representation, we will review previously the main aspects of the knowledge involved in storytelling systems.

Narrative has different aspects in terms of representation (Gervás and León, 2014), each of which has a different natural structure. Every story generation system focuses in a subset of these aspects and holds them by means of a certain set of data structures that represents the system knowledge. For example, Brutus (Bringsjord and Ferrucci, 2000) and

Minstrel (Turner, 1993) emphasised the thematic aspect of the narrative, that is the central topic a text treats. Brutus main theme was betrayal, while every Minstrel story started on a moral that was used as the initial seed.

Still on this subject, another relevant conclusion mentioned by (Gervás and León, 2014) is that the same information may be represented through different data structures without affecting its essence, or a data structure can be extended for representing additional types of information. For example, Brutus (Bringsjord and Ferrucci, 2000) used a specific representation for representing the betrayal. Brutus was developed using a logic-programming system called FLEX, which is based on the programming language Prolog. Its knowledge about betrayal was modelled by a set of statements in FLEX, called *frames*. Every frame formalized the essential characteristics of betrayal: the betrayer, the betrayed, the locations, the actions involved, etc. Mexica (Perez y Perez, 1999) used a wider representation of the relationships between the characters, not specifically focused on betrayal. Relations in Mexica are of two types: emotional links and tensions. Emotional links represent affective reactions between characters. They are defined in terms of three attributes: type (love or friendship), valence (positive or negative) and intensity. Tensions represent if there is a conflict between two characters. It is defined by a type (of conflict) and a state (on or off).

In both examples the same narrative aspect is represented differently in every system, but it can be conceptually identified as a shared concern.

3.1 Dimensions of the narrative

For the purpose of this paper, we are considering a previous work of (Gervás and León, 2014), who analysed the most relevant classifications of the story generation systems according to the knowledge they managed, and proposed their own list of suitable dimensions obtained from the different aspects of a narrative:

- The discourse sequence aspect: a sequential discourse of conceptually conveyed items.
- The simulation aspect: a representation of the activity of agents in terms of actions, interactions, mental states, and movement between locations.
- The causal aspect: a structured representation of causal relations between elements in the story.
- The intentional aspect: a representation of the motivations of agents.
- The thematic aspect: a representation of the theme of parts of the story.
- The emotional aspect: a representation of the emotions involved in or produced by the story.
- The authorial aspect: a representation of the intentions of the author.
- The narrative structure aspect: representations of the story in terms of narratological concepts of story structure.

3.2 Considerations for grammar definition

In addition to these semantic aspects, the proposed CNL grammar definition should meet the common requirements expressed by (Kuhn, 2010):

- Concreteness: CNL grammars should be fully formalized and interpretable by computers.
- Declarativeness: CNL grammars should not depend on a concrete algorithm or implementation.
- Lookahead Features: CNL grammars should allow for the retrieval of possible next tokens for a partial text.
- Anaphoric References: CNL grammars should allow for the denition of nonlocal structures like anaphoric references.
- Implementability: CNL grammars should be easy to implement in different programming languages.
- Expressivity: CNL grammars should be suffciently expressive to express CNLs.

One of the major challenges that faces the target representation is to provide a unambiguous formalism while keeping Natural Language expressiveness.

4 A proposed representation for the narrative dimension

The dimension considered firstly in the CNL grammar is the narrative aspect. It focuses on identifying the underlying structure of the narrative, understood as the framework that supports the inner consistency of the story. From a procedural point of view, the narrative aspect defines the actions performed in order to enhance this skeleton, providing a progressively enriched narrative as a result. This dimension can be traced in the knowledge representation of several of the referred systems (Meehan, 1977b; Dehn, 1981; Turner, 1993; Perez y Perez, 1999; León and Gervás, 2014)

As noted by (Gervás and León, 2014), a different fundamental aspect of narrative is the fact that it can be analyzed in terms of recurring structures that articulate its main ingredients into abstractions that allow its description at a higher level than simple enumerations of events. Along this same line, Propp work (Propp, 1968) is an effort for systematizing the representation of this aspect.

Lang works provided a very interesting step forward to this matter (Lang, 1999) by developing a declarative model for simple narratives. This model described stories in terms of a sequence of events, trying to provide a combined response to the two traditional approaches: declarative and procedural. In the declarative approach the generated text fits a structure that has been defined before (Rumelhart, 1975). By contrast, in the procedural approach, the text was modelled according to a creation process that emulated human authors (Lebowitz, 1985; Turner, 1993).

4.1 Story structure

The proposed structure for representing stories is conceptually based on previous work (Lang, 1999), in the sense of a story is composed by a setting and an episode list, which both have temporal intervals associated with them.

Every episode can be expressed as a N-tuple composed of four elements:

- An initiating event
- An emotional response on the part of the protagonist
- An action response on the part of the protagonist
- An outcome or state description which holds at the conclusion of the episode

4.2 Vocabulary definition

The vocabulary provides the terms for sustaining the conceptual model of every specific dimension. Each dimension can be considered as a domain itself, understood as a unit composed by a cohesive set of interconnected concepts. These concepts are provided by a collection of domain terms and their relations. So, they are the building units for expressing the knowledge relevant for the considered dimensions. In order to formalize this structure, the vocabulary is defined as follows:

- A *term* designates a significant knowledge entity that can be represented by a common noun or a noun phrase.
- A *name* designates unambiguously a significant entity that represents a single thing. It is typically a proper name, referring a character, a place, an object, etc.
- A *verb* designates a relationship, situation, or action involving one or more terms or names. The verbs are both the richest and the most complex elements of the vocabulary. A verb can be expressed in an active or a passive form. Verbs can also be qualified by modal verbs, so they can communicate probability, ability, permission, obligation and advice.
- An *adverbial* serves for expressing the circumstances involving the action defined by the verb. It is an optional part of the sentence.

4.3 Grammar definition

The expression *Subject* + *Verb* defines an attribution or a state related to the Subject, that is a placeholder for a Term or a Name.

The combination *Subject* + *Verb* + *Object* defines a semantic relationship and has two placeholders filled by Terms/Names. The particular case of the verb to be must be considered as a typical expression for building descriptions.

The combination *Subject* + *Verb* + [*Adverbial*] + *Object* defines an action performed over an object. The action defined by the verb can be better put into context by means of adverbials. These can be used for expressing the circumstances in which action takes place.

Sentences can be combined in order to create compound sentences or subordinate clauses.

The sentence will be expressed in a declarative manner. For example, the following statement shows a complete case:

The main character finds accidentally a clue that allows him to finish his research.

The CNL grammar defines a collection of syntax rules and constrains for representing the knowledge as statements. It presupposes the existence of a vocabulary because it addresses the *terms* and *verbs* defined in the vocabulary. The general structure of every statement is composed of four parts: the initiating event, an optional part that expresses a change in the subject emotions, another optional part that expresses the actions taken by the subject, and a ending sentence that expresses the outcome.

So far, we have presented the general pattern of the grammar. It is a set of rules which allow going from a simple to a reasonably complex structure. This last point can be reached by means of connectives. The noun phrases can also be combined and qualified using different quantifications and prepositional phrases, but always with the certainty that it will produce sentences that are grammatically correct.

5 A proposed representation for the simulation dimension

Another relevant aspect of narrative is the representation of characters, their behaviour, and the expression of their mental state, their relations with one another, their motivations, and their beliefs. The simulation aspect has been frequently highlighted as the leitmotiv for the representation of narratives in some approaches to story generation (Lebowitz, 1985; Bringsjord and Ferrucci, 2000). Such approaches usually focus on representing characters and rules that may govern their behaviour and interaction.

5.1 Modelling the affinity

The simulation aspect refers to the characterization of the persona in terms of the interaction between each other. That is, this aspect covers a wide scope that ranges from the definition of characters attributes and traits, to the delimitation of their affinities. Naturally, this also relates to the way in which the characters interact with each other and the actions they perform motivated by the result of such interactions. The affinity aspect have been studied by several authors and systems (Imbert and De Antonio, 2005; Si et al., 2006; Méndez et al., 2016). The present work is related to the system developed by (Méndez et al., 2016). Usually, the authors apply an affinity factor for modelling the way in which social interaction affects the behaviour of the characters with each other. In other cases, affinity is affected by other factors, such as social obligations and characters goals. An additional aspect of affinity to keep in mind is that it is not symmetrical. Given two characters, their mutual affinity is likely to be dissimilar.

There are several possible ways for expressing the affinity between two characters. A first option is the use of a collection of symbolic values that allow reasoning about them and the ongoing simulation, but that difficult the operation. On the other hand, the use of numeric values makes easier operating with them, but hinders understanding the evolution of the simulation.

With a view to representing the affinity in terms of Natural Language, the simpler choice is to use a collection of adjectives that represent a range of numeric values.

In the referred model (Méndez et al., 2016), authors have modelled additionally four levels of affinity according to four different kinds of affinity: *foe* (no affinity), *indifferent* (slight affinity), *friend* (medium affinity) and *mate* (high affinity). These values can be suitable for expressing it in a first approach.

5.2 Vocabulary definition

As stated previously, the vocabulary provides the terms for defining the model of the corresponding dimension. In the domain of the simulation, the vocabulary is defined as follows:

- A *term* designates a significant knowledge entity that can be represented by a common noun or a noun phrase. It is exactly as defined in the context of narrative structure.

- A *name* designates unambiguously a significant entity that represents a single thing. It is exactly the same entity as in the case of the narrative structure.

- A *verb* states a feature, a trait, or an action involving one or more terms or names. The verb in the simulation aspect provides basically a definition or an action performed by a character

5.3 Grammar definition

The simulation aspect is defined in terms of characters' traits and interactions. These special features need a specific way of being represented.

In this regard, the CNL created for expressing all these dimensions will contain basically sentences for describing traits and attributing features. It will also be a language for describing actions and interaction. So, the grammar for formalizing it is composed of expressions of the following kind: *Subject + Verb + Attribute* or: *Subject + Verb + Object*.

In the first case, the expression reflects the definition of a trait or a feature. The *attribute* will be represented by means of a *term*. The *verb* will typically be the *to be* and *to have*.

In the second case, the verb expresses an action. The character, that is the subject, performs some action that affects something or someone. So, the *object* can be either a *term* or a *name*.

This last type of expression can also be used for defining the affinity between characters, so there will be sentences like: *John is a friend of Mary*.

6 Conclusions and future work

This paper proposed the application of a CNL for eliciting and exchanging knowledge between story generation systems as a means of collaborative generation of stories. It also discusses a model for generating this CNL automatically from different knowledge representation formalisms. As explained above, there have been precedents of the use of CNL in the interactive storytelling domain with satisfactory results.

The aim of the proposed representation is to help bridging the variety of knowledge representation in a simple and formal way. The proposed syntax has been defined by a formal grammar but the resulting expressions keep a human-friendly nature.

In this paper, the developed work is centred on two dimensions of the knowledge: the story structure dimension and the simulation dimension. This is just one part of the needed multi-aspectual representation. As mentioned above, there are some other dimensions that must be addressed in future versions of the CNL: the authorial aspect, the emotional aspect, the intentional aspect and the theme aspect.

The future work will be focused on completing the set of the grammar generation rules for expressing these remaining aspects of knowledge involved in storytelling. This work will provide a completely expressive representation that hold co–creation between storytelling systems.

Benchmarking this work can really be complex, and will probably involve a shared effort with other research groups. So, we are working on proposing a shared task in which to compare the quality of stories generated by different systems using the same initial knowledge base. Collaborators will be provided with a grammar definition of a CNL that represents the narrative aspects mentioned in the previous section, along with a set of initial situations written using this grammar from which to generate different stories using the same CNL. They will be required to use as much information as possible in order to generate rich stories that cover one or more of the previously described narrative aspects. These stories must be expressed in the same CNL used to describe the initial situation so that, hypothetically, the output of a system might feed another system in order to provide more details about some of the aspects that may have been left uncovered by previous generators, following a co–creation process where a system can strengthen the weaknesses of another. The outcome of this collaborative process is expected to provide the means to develop an enhanced story creation model.

Aknowledgements

This work was partially supported by the projects WHIM (611560) and ConCreTe (611733), funded

by the European Commission under FP7, the ICT theme, and the Future Emerging Technologies (FET) program.

References

Guntis Barzdins. 2014. Framenet cnl: A knowledge representation and information extraction language. In *International Workshop on Controlled Natural Language*, pages 90–101. Springer.

John A Bateman. 1997. Enabling technology for multilingual natural language generation: the kpml development environment. *Natural Language Engineering*, 3(01):15–55.

Selmer Bringsjord and David A Ferrucci. 2000. Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine. *Computational Linguistics*, 26(4).

Frederik Van Broeckhoven and Olga De Troyer. 2013. Attac-l: A modeling language for educational virtual scenarios in the context of preventing cyber bullying. In *2nd International Conference on Serious Games and Applications for Health*, pages 1–8. IEEE, May.

Frederik Van Broeckhoven, Joachim Vlieghe, and Olga De Troyer. 2015. Using a controlled natural language for specifying the narratives of serious games. In *8th International Conference on Interactive Digital Storytelling, ICIDS 2015*, pages 142–153.

Lynne J Cahill, Christy Doran, Roger Evans, Rodger Kibble, Chris Mellish, Daniel S Paiva, Mike Reape, Donia Scott, and Neil Tipper. 2000. Enabling resource sharing in language generation: an abstract reference architecture. In *LREC*. Citeseer.

MacGregor Campbell. 2011. A new way to play: Make your own games. *New Scientist*, 211(2829):21.

Brian Davis, Pradeep Varma, Siegfried Handschuh, Laura Dragan, and Hamish Cunningham. 2009. Controlled natural language for semantic annotation. In *The Semantic Web: Research and Applications*, pages 816–820. Springer.

Natlie Dehn. 1981. Story generation after tale-spin. In *IJCAI*, volume 81, pages 16–18.

Norbert E Fuchs, Uta Schwertel, and Rolf Schwitter. 1998. Attempto controlled englishnot just another logic specification language. In *Logic-based program synthesis and transformation*, pages 1–20. Springer.

Norbert E Fuchs, Kaarel Kaljurand, and Tobias Kuhn. 2008. Attempto controlled english for knowledge representation. In *Reasoning Web*, pages 104–124. Springer.

Adam Funk, Valentin Tablan, Kalina Bontcheva, Hamish Cunningham, Brian Davis, and Siegfried Handschuh. 2007. *Clone: Controlled language for ontology editing*. Springer.

P. Gervás and C. León. 2014. The need for multi-aspectual representation of narratives in modelling their creative process. In *5th Workshop on Computational Models of Narrative*, OASIcs-OpenAccess Series in Informatics.

P. Gervás. 2012. Story generator algorithms. In *The Living Handbook of Narratology*. Hamburg University Press.

Ricardo Imbert and Angélica De Antonio. 2005. An emotional architecture for virtual characters. In *Virtual Storytelling. Using Virtual Reality Technologies for Storytelling*, pages 63–72. Springer.

Tobias Kuhn. 2009. *Controlled English for knowledge representation*. Ph.D. thesis, Faculty of Economics, Business Administration and Information Technology of the University of Zurich.

Tobias Kuhn. 2010. Codeco: A practical notation for controlled english grammars in predictive editors. In *Controlled Natural Language*, pages 95–114. Springer.

Raymond Lang. 1999. A declarative model for simple narratives. In *Proceedings of the AAAI fall symposium on narrative intelligence*, pages 134–141.

Michael Lebowitz. 1985. Storytelling and generalization. In *Seventh Annual Conference of the Cognitive Science Society*, pages 100–109.

Carlos León and Pablo Gervás. 2011. A top-down design methodology based on causality and chronology for developing assisted story generation systems. In *Proceedings of the 8th ACM conference on Creativity and cognition*, pages 363–364. ACM.

Carlos León and Pablo Gervás. 2014. Creativity in story generation from the ground up: Nondeterministic simulation driven by narrative. In *5th International Conference on Computational Creativity, ICCC*.

Hugo Liu and Push Singh. 2002. Makebelieve: Using commonsense knowledge to generate stories. In Rina Dechter and Richard S. Sutton, editors, *AAAI/IAAI*, pages 957–958. AAAI Press / The MIT Press.

James R. Meehan. 1977a. Tale-spin, an interactive program that writes stories. In *In Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pages 91–98.

James R Meehan. 1977b. Tale-spin, an interactive program that writes stories. In *IJCAI*, volume 77, pages 91–98.

Gonzalo Méndez, Pablo Gervás, and Carlos León. 2016. On the use of character affinities for story plot generation. In *Knowledge, Information and Creativity Support Systems*, pages 211–225. Springer.

R. Perez y Perez. 1999. *MEXICA: A Computer Model of Creativity in Writing*. Ph.D. thesis, The University of Sussex.

Richard Power, Robert Stevens, Donia Scott, and Alan Rector. 2009. Editing owl through generated cnl.

Vladimir Propp. 1968. Morphology of the folk tale. 1928.

A. Reed. 2010. *Creating Interactive Fiction with Inform 7*. Cengage Learning.

Ehud Reiter, Robert Dale, and Zhiwei Feng. 2000. *Building natural language generation systems*, volume 33. MIT Press.

Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, and Yasmin Kafai. 2009. Scratch: Programming for all. *Commun. ACM*, 52(11):60–67, November.

David E Rumelhart. 1975. Notes on a schema for stories. *Representation and understanding: Studies in cognitive science*, 211(236):45.

Roger C. Schank and Robert P. Abelson. 1975. Scripts, plans, and knowledge. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'75, pages 151–157, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Rolf Schwitter. 2010. Controlled natural languages for knowledge representation. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1113–1121, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mei Si, Stacy C Marsella, and David V Pynadath. 2006. Thespian: Modeling socially normative behavior in a decision-theoretic framework. In *Intelligent Virtual Agents*, pages 369–382. Springer.

P. Singh, T. Lin, E. T. Mueller, G. Lim, T. Perkins, and W. L. Zhu. 2002. Open mind common sense: Knowledge acquisition from the general public. In *On the move to meaningful internet systems 2002: Coopis, DOA and Odbase*, pages 1223–1237. Springer.

Scott R. Turner. 1993. *Minstrel: A Computer Model of Creativity and Storytelling*. Ph.D. thesis, University of California at Los Angeles, Los Angeles, CA, USA. UMI Order no. GAX93-19933.

3.4. Artículo: A Challenge Proposal for Narrative Generation using CNLs

Este artículo propone un *challenge* a la comunidad de la generación automática para representar el conocimiento de sus sistemas en forma de CNL. El objetivo del mismo es ampliar la base de ejemplos en cuanto a mecanismos de representación entre cuantos sistemas fuera posible.

3.4.1. Cita completa

E. Concepción, G. Méndez, P. Gervás, and Carlos León (2016). *A Challenge Proposal for Narrative Generation Using CNLs*. In Proceedings of the 9th International Natural Language Generation conference, pages 171–173, Edinburgh, UK. Association for Computational Linguistics.

3.4.2. Resumen original

We propose a competitive shared evaluation task for Narrative Generation. It would involve the generation of new stories for a given domain from common ground knowledge shared by all systems. A set of source materials will be provided for development, represented in Controlled Natural Language (CNL), which should also be used to phrase the text outputs of participating systems. By having all participating systems operate from the same sources for knowledge and generate in a compatible output format, comparability of the results will be enhanced. Submitted results will be subject to both automatic and human evaluation.

A Challenge Proposal for Narrative Generation Using CNLs

Eugenio Concepción and **Gonzalo Méndez** and **Pablo Gervás** and **Carlos León**
Facultad de Informática
Instituto de Tecnología del Conocimiento
Universidad Complutense de Madrid
{econcepc,gmendez,pgervas,cleon}@ucm.es

Abstract

We propose a competitive shared evaluation task for Narrative Generation. It would involve the generation of new stories for a given domain from common ground knowledge shared by all systems. A set of source materials will be provided for development, represented in Controlled Natural Language (CNL), which should also be used to phrase the text outputs of participating systems. By having all participating systems operate from the same sources for knowledge and generate in a compatible output format, comparability of the results will be enhanced. Submitted results will be subject to both automatic and human evaluation.

1 Introduction

A story generator algorithm (SGA) refers to a computational procedure resulting in an artefact that can be considered a story (Gervás, 2012). The term `story generation system` can be considered as a system that applies a SGA to construct stories. There is a growing population of such story generation systems that share two significant characteristics: one, they operate from a set of knowledge resources that act as input to the story generation process; two, they rely on elementary text building solutions – usually based on template filling – for producing human-readable versions of their outputs. Comparative evaluations of any kind between these story generation systems are very difficult because: different systems start from different (unrelated) knowledge resources, and text outputs of the different systems are heavily influenced by the (different) sets of templates employed to render them. A common approach to acquiring knowledge resources is to mine a set of reference stories, to obtain from them the required knowledge. These resources usually make explicit two types of information that is implicit in the stories: relation between events in the story and latent variables relevant to it – such as causality, emotion, affinities between characters, narratological concepts... –, and/or information about typical/acceptable sequencing between events – depending on the degree of refinement of the system, sometimes based on the latent variables.

The present proposal revolves around the idea of developing a Controlled Natural Language (CNL) that can be used to specify the *source material* for a story generation task. A CNL is an engineered subset of natural languages whose grammar and vocabulary have been restricted in order to reduce both ambiguity and complexity of full natural languages (Schwitter, 2010). If such a CNL could be used to represent a set of reference stories, while ensuring that any latent variables are made explicit in the representation, it should be possible to automatically extract the relevant knowledge resources from such source material. To make this possible, the type of source material required should include a set of example stories either enriched with explicit mentions of latent variables or accompanied by explicit declaration of the relation between elements in the stories and the latent variables. If textual outputs of story generation systems could be phrased in such a CNL, it should be feasible to compare outputs of different systems on a shared common footing.

Proceedings of The 9th International Natural Language Generation conference, pages 171–173,
Edinburgh, UK, September 5-8 2016.

2 Conceptual Basis

In (Gervás and León, 2014) the authors provided a list of the most relevant classifications of the story generation systems according to the type of knowledge resources that they rely on, and the way these knowledge resources are implemented as specific data structures. That paper proposed a list of aspects of a narrative relevant to story telling systems in this sense: including the discourse produced for the story, the representation of the activity of agents – in terms of actions, interactions, mental states, and movement between locations –, the causal relations between elements in the story, the motivations of agents, the theme of parts of the story, the emotions involved in or produced by the story, the intentions of the author, and the narratological concepts involved in the story structure. These various aspects constitute sources of candidate features for the role of latent features relevant for story telling.

CNLs can be considered as a tradeoff between the expressivity of natural languages and the need for a formal representation that can be handled by computers. The requirements for the definition of a CNL grammar (Kuhn, 2010) relevant for the present purpose are: that it should be fully formalized and interpretable by computers, it should not depend on a concrete algorithm or implementation, it should be easy to implement in different programming languages, and it should be sufficiently expressive (for the task at hand).

3 A Proposal for a Story Generation Shared Task

The feasibility of the shared task relies on the development of two basic resources: a grammar for a CNL capable of representing the various aspects relevant to story telling and the resources required by story generation systems, and a set of source materials that encode the necessary knowledge for generating stories in a specific domain covering a selected set of the relevant aspects.

The challenge as proposed is addressed to existing story generation systems.[1] The task would involve: extracting task-specific instances of the knowledge resources required for the candidate system from the source materials provided, adapting the text rendering modules of the story generation system to generate stories as close as possible to the the CNL developed for the task, and submitting the resulting stories for evaluation.

3.1 Development of Resources

The proponents of the challenge intend to enlist the collaboration of authors of existing story generation systems with a two-fold purpose: to ensure that the developed resources provide coverage of as many aspects of narrative deemed relevant from a computational perspective, and to raise interest in the challenge and build a community of candidate participants. The collaboration envisaged would take the form of providing sample instances of the knowledge resources employed by their system for generation in a domain of their choice.

3.1.1 The CNL

Such resources will be used to inform the iterative development of the grammar for the CNL. An initial grammar will be built covering aspects common to all systems and all resources. This grammar will be progressively enriched with any additional aspects covered by some systems and not by others, until all selected aspects are covered. Depending on what aspects are covered by the compiled resources and how easy they are to embed into a story, decisions will need to be made on how to represent the relevant latent variables, either as explicit enrichment of stories or as separate declaration of their relation to story elements. Some progress has already been made along these lines (Concepción et al., 2016).

A parser will be developed for the CNL, capable of building actual data structures for the various aspects represented. Both the grammar for the CNL and the code for the parser will be made available to participants. The parser will be designed so that it has a specific module for saving the data structures to disk. Such module may be reinstantiated by participants to select which part of the knowledge in the data structure is saved onto what particular representation format for a particular system.

[1] It may be undertaken by researchers willing to develop a system from scratch if they consider it feasible, but the effort involved would be much higher.

3.1.2 The Source Materials

The CNL will be used to develop source materials for a particular domain chosen as focus for the challenge. Source materials may consist of a set of enriched stories and/or a set of definitions of relations between story elements and latent variables. Additional knowledge relevant to the domain may also need to be encoded – using the CNL – in the source materials. The basic scope and structure of such additional material will be based on the concept of a *story bible* or *show bible* as considered by screenwriters for information on a television series' characters, settings, and other elements.

3.1.3 Evaluation Procedures

Textual outputs produced by participating systems will be processed using the parser described in 3.1.1. Outputs will be rated automatically on the following parameters: grammaticality – based on conformance to the grammar –, novelty with respect to reference stories in the source materials – data structures built by the parser from the outputs will be compared with those arising from the reference stories according to existing metrics for narrative similarity (Peinado et al., 2010; Hervás et al., 2015) – , and additional rating schemes developed for any relevant features – as the data structures generated by the parser will include explicit representation of these aspects, development for specif metrics is possible for features like degree of causal connectivity, rise and fall of emotion or affinity between characters over a story, or any others explicitly represented.

For the parameters chosen, judgements from human evaluators will also be compiled.

3.1.4 Expected Timeline

A tentative timeline is proposed which would involve: requesting contributions – as samples of knowledge resources – from interested researchers by the end of September 2016, publish source materials in March 2017, outputs to be submitted by participants by July 2017, final results presented at INLG 2017. However, in view of the various uncertainties existing in the proposal, it may be necessary to comtemplate the need to postpone the submission deadline to 2018, in which case the tentative timeline may be re-distributed accordingly over the intervening period.

4 Expected Benefits

The development of agreed versions of source materials from which story generation resources can be extracted, a grammar for outputs of story systems, and procedures for quantitative measurement of relevant features would constitute significant benefits.

Acknowledgments

This proposal has been partially supported by projects WHIM (611560) and PROSECCO (600653) funded by the European Commission, Framework Program 7, the ICT theme, and the Future Emerging Technologies FET program, and by project IDiLyCo (MINECO/FEDER TIN2015-66655-R), funded by the Spanish Ministry of Economy and the European Regional Development Fund.

References

E. Concepción, P. Gervás, G. Méndez, and C. León. 2016. Using CNL for knowledge elicitation and exchange across story generation systems. In *5th Workshop on Controlled Natural Language (CNL 2016)*, Aberdeen, Scotland, 07/2016. Springer, Springer.

P. Gervás and C. León. 2014. The need for multi-aspectual representation of narratives in modelling their creative process. In *5th Workshop on Computational Models of Narrative*, OASIcs-OpenAccess Series in Informatics.

P. Gervás. 2012. Story generator algorithms. In *The Living Handbook of Narratology*. Hamburg University Press.

R. Hervás, A. Sánchez-Ruiz, P. Gervás, and C. León. 2015. Calibrating a metric for similarity of stories against human judgment. In *Creativity and Experience Workshop, International Conference on Case-Based Reasoning*, Bad Homburg, Frankfurt, Germany, 09/2015.

T. Kuhn. 2010. Codeco: A practical notation for controlled english grammars in predictive editors. In *Controlled Natural Language*, pages 95–114. Springer.

F. Peinado, V. Francisco, R. Hervás, and P. Gervás. 2010. Assessing the novelty of computer-generated narratives using empirical metrics. *MINDS AND MACHINES*, 20(4):588, 10/2010.

R. Schwitter. 2010. Controlled natural languages for knowledge representation. In *Proc. of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 1113–1121, Stroudsburg, PA, USA. Association for Computational Linguistics.

3.5. Artículo: A common model for representing stories in automatic storytelling

Este artículo presenta el primer modelo sistematizado de representación, contando con los elementos presentados en la introducción del presente capítulo. El artículo tiene una contrapartida en la parte de arquitectura (Concepción et al., 2017a), pues ambos modelos están orientados a la articulación de un marco para la construcción de sistemas generadores de historias. El planteamiento expuesto en el modelo de representación se enfoca en mantener la flexibilidad de la representación, de forma que permita crear instancias de historia donde no todos los aspectos, o dimensiones, estén cubiertos; y también permita agregar elementos a la representación sin tener que modificar su esquema. El primer aspecto guarda nuevamente relación con la forma de operar del modelo arquitectónico propuesto en su contrapartida, puesto que en cada componente puede completarse una parte del contenido de la representación total de la historia que se está generando.

3.5.1. Cita completa

E. Concepción, Gervás, P., and Méndez, G. (2017). *A common model for representing stories in automatic storytelling*, in 6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017, Madrid, Spain.

3.5.2. Resumen original

The present paper proposes a common representation model that allows the free exchange of knowledge between different story generation systems as a base for a collaborative environment to run an enhanced process of literary creation. In addition to this objective, this model aims at the development of a story representation formalism for creating a common knowledge base that can be fed in the future with the outcomes of new storytelling systems, without the need to adapt it to every system-specific representation model.

A Common Model for Representing Stories in Automatic Storytelling

Eugenio Concepción[1], Pablo Gervás[2], and Gonzalo Méndez[2]

[1] Facultad de Informática
[2] Instituto de Tecnología del Conocimiento
Universidad Complutense de Madrid
econcepc@ucm.es, pgervas@sip.ucm.es, gmendez@fdi.ucm.es

Abstract. The present paper proposes a common representation model that allows the free exchange of knowledge between different story generation systems as a base for a collaborative environment to run an enhanced process of literary creation. In addition to this objective, this model aims at the development of a story representation formalism for creating a common knowledge base that can be fed in the future with the outcomes of new storytelling systems, without the need to adapt it to every system-specific representation model.

Keywords: computational creativity, story generation systems, knowledge representation, formal languages

1 Introduction

Automatic story generation is a part of a wider research area in Artificial Intelligence named Computational Creativity (CC), which is the pursuit of creative behaviour in machines [28]. A story generator algorithm (SGA) refers to a computational procedure resulting in an artefact that can be considered a story [9]. The term story generation system can be considered as a synonym of storytelling systems, that is, a computational system designed to tell stories.

Story generation systems are faced with a significant challenge of acquiring knowledge resources in the particular representation formats that they use. They face an inherent difficulty when using formal languages in the detachment between the formulation of the needs in the real world and its representation in a formal construction. These difficulties are increasingly greater when the systems participate in a co-creation model with humans [6]. The co-creation process in the context of Computational Creativity implies the involvement of several systems and humans working together in an iterative cycle of enhancement.

The present paper introduces a model for expressing knowledge related to the domain of narrative. The goal of this formalism is to ease the interchange of information between different story generation systems operating in a collaborative ecosystem. Its purpose is to establish a co-creation cycle inside an architectural model that allows the involved systems to take advantage of the shared knowledge model and use it for enhancing the quality of the generated texts.

2 Background

This section reviews the way story generation systems represent the knowledge they need to create stories. The existence of commonalities across the existing systems is necessary to establish a shared model for knowledge representation.

TALE-SPIN [17] was basically a planning solver that wrote up stories by narrating the steps performed for achieving the characters' goals. TALE-SPIN stories were set in a forest. The system took as input a collection of characters with their corresponding goals, and generate a sequence of steps while resolving them. After that, it wrote up a story narrating the steps performed for achieving those goals. TALE-SPIN knowledge representation relied heavily on Conceptual Dependency Theory [25], and it represented the problem domain using a set of primitives, expressed in a formal language.

Author [7] was the first story generator to include the author intentionality as a part of the story generation process. Dehn considered that stories were a retrospective justification for a plot conceived in authors mind. For this reason, Author tried to emulate a real writer's mind. The process considered that there were underlying objectives driving the storytelling process, even when the author had not clearly expressed them. From an architectural point of view, Author was a planner, but, unlike TALE-SPIN, it used the planning to fulfill authorial goals instead of character goals. During story generation, Author built iteratively the plot for better meeting the goals of the author, that could also be readjusted in every iteration after a reviewing stage.

Mexica [21] was developed as a computer model whose purpose was studying the creative process. It generated short stories about the early inhabitants of Mexico. Mexica was a pioneer in that it took into account emotional links and tensions between the characters as a means for driving and evaluating ongoing stories. Mexica knowledge management could be considered complex and sophisticated. Its knowledge base included several types of structures for representing things like characters relationships, actions, emotional links, and a literary base composed of previously generated stories.

Thespian [26] introduced a richer representation of characters subjectivity by modelling their beliefs. Every character had a representation of their own beliefs about the story world, and those beliefs could even be false. This feature made characters could reason about the way the others see the world.

Fabulist [23] is a complete architecture for automatic story generation and presentation. It combines an author-centric approach together with a representation of characters intentionality, and an open-world planning for creating highly believable stories. Fabulist has been designed to enable story generation with little prior knowledge built into the system [24]. This feature was intended to allow Fabulist to generate stories of types that were not anticipated by the systems creator. Thus, Fabulist includes among its goals plot coherence and character believability when performing story generation.

STellA (Story Telling Algorithm) [15] is a story generation system that mixes a non-constrained simulation-based production of world states and narrative actions as source material for a conceptual space exploration engine. The system

manages states in a non-deterministically generated space of partial stories, making choices until it finds a satisfactory simulation of events progression of the simulations that is rendered as a story.

ProppWryter [10, 11] is a story generation system that creates russian folktales according to Propp's generation rules [22]. These rules provide a very clear description of how the folktales morphology could be used for story generation. It uses a set of abstractions for representing the essential concepts defined by Propp, especially the character function, and defines a procedure that first chooses a sequence of character functions to act as abstract narrative structure to drive the process, and then progressively selects instantiations of these character functions in terms of story actions to produce a conceptual representation of a valid story.

Charade [18] is an affinities-based story generation system which generates stories after simulating the evolution of a relationship between two characters using their mutual affinities, and operating as decoupled as possible from the story domain. This system is an agent-based architecture developed using JADE.

In addition to these single storytelling systems, Slant [20] must be considered as an example of storytelling systems interoperating for producing a enhanced outcome. It is a complete architecture for creative story generation that integrates different types of story generation systems. Its architecture is the result of the integration of several different components: one based on Mexica [21], one based on GRIOT [13] , and a new system specifically developed for Slant.

Slant represents the story in a shared data structure, according to the blackboard architectural pattern. This resource allows the participating storytelling systems and components to create stories collaboratively. The goal is to allow the systems to influence each other for generating an enhanced result. The blackboard architecture and the Slant story XML format that is used, open up new possibilities for collaboration between creative literary systems, allowing models of creativity to be developed and added in different configurations.

The process for generating stories in Slant begins with minimal, partial proposals from a simple unit, named the Seeder [20]. In turn, the subsystems MEXICA, Verso, and Fig-S read and complete the set of proposals, each according to its focus. When they have finished the processing, the enhanced story specification is sent to GRIOT-Gen so conceptual blending can be applied to the relevant templates and then to the text generation component of Curveship. Finally, Curveship-Gen generates a finished story in natural language, delivered as a text file that can be read and considered by human readers.

3 Statement of the problem

To enable a better understanding of the problem, let us consider how to develop an automatic story generation system that could generate a story like "War and peace" [27]. This novel tells the story of the Napoleonic invasion of Russia and its impact on Russian society through the stories of five Russian aristocratic families. The novel is a truly monumental work, containing several chapters de-

voted to philosophical disquisitions, detailed military strategy scenes, and a rich description of the emotional evolution of the characters. From an anthropological perspective, it provides a complete historical and cultural context.

In the light of the above, creating such a system would be a formidable challenge. Considering the capabilities of the existing storytelling systems, the target generator should be able to operate at different levels of detail and resolve many interwoven subplots. For example, when generating the narration of the war, the system should create a detailed description of the movements carried out by the Napoleonic army on Russian territory, and the corresponding response of the Russian generals. To do this, it would need to focus on strategy, with a discourse aimed at describing the movements of troops, the relationships between the actions of the Napoleonic army and the countermeasures of the Russian army, the activity of the convoys of military supplies, the terrain difficulties, and finally, a detailed description of the battles fought during the campaign. In such type of narration there is no room for long dialogues, nor for focusing on individual thoughts or feelings. Narrative resources are aimed at representing the complete picture of events. The detail is used in the description of the facts and the scenarios in which they take place.

On the other hand, to develop the story of the relationships between the characters that take place during the war, it would require the system to be able to simulate the various characters involved, their feelings, their intentions, their social norms and their actions. The interaction between the characters would lead to changes in mutual perception and feelings. To convey all this evolution, it would be necessary to resort to witty dialogues, in which the characters reveal their feelings and intentions, or to minutely narrated scenes that show how they behave. Of course, a believable love story can only be achieved by employing a good amount of knowledge about every single character. This knowledge includes a complete psychological portrait, which allows a deep understanding of its motivations, a proper representation of the historical context, and many other aspects related to the characters' consistency.

The above examples do not exhaust the list of requirements. In a novel like "War and peace" there are multiple aspects that would require of a specific processing in a storytelling system. To mention some of them: the thematic aspects require a simulation model centred on the intentions of the author; the description of a duel requires a simulator of physical world; and a philosophical reflection about the status of the servants requires a reasoning engine that includes knowledge concerning the social context in nineteenth-century Russia.

It seems quite difficult to generate such a novel with a single storytelling system. The prior example just tries to pose a question about the complexity of working at different levels of knowledge representation. The simplest way of creating rich and complex stories is using different systems, generating different types of content according to their capabilities.

The existence of different story generators, each with a different approach not only to how to identify the next action to continue an existing draft, but also to how to evaluate the quality of a partial draft requires the definition of

a specific interaction protocol to guide the way they collaborate among them to create a single consensus draft. The context of the interaction would be a set of story drafts that are being developed over a sequence of interactions. As such, every draft contains a fusion of the information provided by the various generators, represented in the shared format.

4 Methodology

From a methodological point of view, this paper focuses on identifying the dimensions considered by the knowledge managed by storytelling systems. The strategy being followed was originally outlined in a previous paper [5]. A set of dimensions was selected for identifying the common aspects in the representation of knowledge in storytelling systems.

The representation model is strongly influenced by the components of narrative identified in the classic Narratology [2] [1]. Narratology is a humanities discipline dedicated to the study of the logic, principles, and practices of narrative representation [16]. Its concepts and models are widely used as heuristic tools, and narratological theorems play a central role in the exploration and modeling of our ability to produce and process narratives in a multitude of forms, media, contexts, and communicative practices.

In order to clarify the developed model, it is necessary to set out the main concepts that come into play. The following definitions are based on those established by the traditional literature [2] [1]:

Narrative is considered as a story articulated in a discourse. It is a complex concept, which involves several components itself.

Story is the content of the narration; it includes what happens, namely, the plot, and the space in which the action occurs.

Space includes the characters, settings, props and anything which is present either physically or abstractly in the space of the narrative —that is to say, the existents. Because existents change and evolve over time, the space also consists of an initial state which contains the set of all existents' states as they exist before the start of the plot.

Plot is the basic structure of any story. A plot is a set of events with an overall structure which represents both the temporal ordering, and the causal relations between the events. Events typically consist of one or more low-level actions, instigated by and/or affecting a number of entities in the space. The concept of plot encompasses the ways in which the events and characters' actions in a story are arranged and how this arrangement in turn facilitates identification of their motivations and consequences.

Discourse is the particular mean of telling a story. This may include several aspects such as the narrator's perspective, the ordering and duration of the events in the plot, etc.

Apart from these high level concepts, that constitute the roots of the model, there are other relevant concepts that have been considered due to its importance in this representation model. The representation of the characters is one of them.

According to Jannidis [16], there are three forms of relevant knowledge for the narratological analysis of the characters:

- The basic type, which provides a very fundamental structure for those entities which are seen as sentient beings.
- The character models or types —that is to say, character stereotypes.
- An encyclopedic knowledge of human beings behaviour, ranging from everyday situations knowledge to any other relevant knowledge that could contribute to the process of characterization.

The proposed representation model considers these three levels, covering totally the two first, and partially the third one.

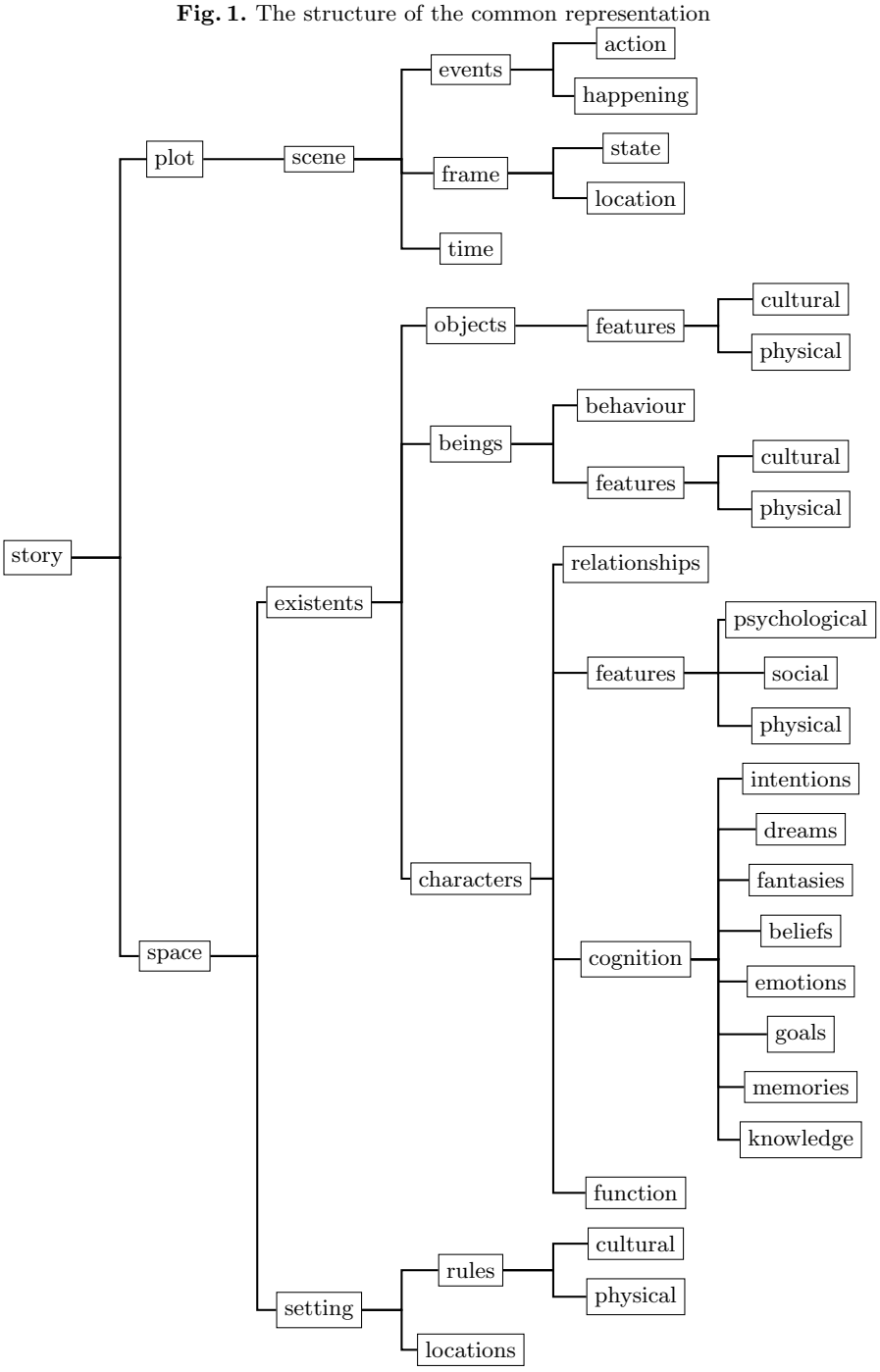
Another key concept is the narrative space, that goes further than the representation of a world in terms of a simple container for existents and a location for events. Resorting again to narratology, we have considered five levels [16] for modelling the narrative space:

- Spatial frames: the immediate surroundings of actual events. Spatial frames are shifting scenes of action, and they may flow into each other. They are hierarchically organized by relations of containment, and their boundaries may be either clear-cut or fuzzy.
- Setting: the general socio-historic-geographical environment in which the action takes place. In contrast to spatial frames, this is a relatively stable category which embraces the entire text.
- Story space: the space relevant to the plot, as mapped by the actions and thoughts of the characters. It consists of all the spatial frames plus all the locations mentioned by the text that are not the scene of actually occurring events.
- Narrative (or story) world: the story space completed by the reader's imagination on the basis of cultural knowledge and real world experience.
- Narrative universe: the world (in the spatio-temporal sense of the term) presented as actual by the text, plus all the counterfactual worlds constructed by characters as beliefs, wishes, fears, speculations, hypothetical thinking, dreams, and fantasies.

5 Proposed model

If we analyse in detail the existing storytelling systems, two conclusions can be reached regarding knowledge representation [5]. On the one hand, the knowledge related to the story generation process is inherently ad hoc, and consequently hardly exportable from one system to another. On the other hand, we can observe that there is a common element for every storytelling system that can be interchanged, the generated story. By means of a careful analysis of the structure of the stories, the common elements arises. The resulting representation model is summarized in Figure 1.

Fig. 1. The structure of the common representation



The model has been designed in a hierarchical structure, in which the root concept is the **story**. A story represents what both intuitively and narratologically can be considered a story, that is, a narration of events happening in a setting. It is composed by the two classic narratological components: the plot and the space.

The **plot** is represented as a sequence of scenes. A **scene** is conceptually related to the division of a play, that represents a single episode inside the plot. It is clearly conditioned by the time division, which means that is a sequence of events that happen during a time frame. From a spatial point of view, it is also constrained to take place in a single spatial frame —considering the spatial frame definition mentioned before. So, the scene is composed by a sequence of **events**, that can be actions or happenings. An **action** is an act performed by one or more characters in the story, generating consequences. The resulting consequences of every action are expressed as a modification in the global state of the space —considering it as the whole setting and the existents. A **happening** is an event that happens in the plot, as an accident or as a consequence of a prior action or happening. A happening can be natural —it rains— or artificial —a car accident.

The **space** encompasses the whole universe in which the plot is developing. It is composed by the setting and the existents. The **existents** are the whole set of actors that take a part in the story. They can be characters, living beings —an animal—, and an object in the setting. The two last types are mainly defined by their physical features and their cultural significance in the story. The **characters** are the most relevant, and also the most complex to represent, elements in the story. The proposed model considers not only their physical, psychological and social features, but also their cognitive-related characteristics. The cognition of the characters is represented in a very detailed manner due to its importance for ensuring story consistency and characters liability. The aspects considered have been chosen after analysing those used by the existing storytelling systems [26, 15, 7, 14, 19, 21] and theoretical studies about Narrative [1, 16]. So, the representation of cognition includes the following facets:

- Goals: The goals are the results or achievements toward which the character effort is directed. The model considers two types of goals: conscious and unconscious. In the first case, the character is aware of them, in the second, they drive the character's actions, but he/she is not aware of them.
- Intentions: The intentions refer to the general plan that every character has, and the drive for his/her actions.
- Knowledge: Despite the characters act and interact in the same space, every single character could have different levels of knowledge concerning it. That means that the characters are not considered to be omniscient. This knowledge can evolve over the time, so characters can be acquiring or discarding knowledge as the story develops.
- Memories: Unlike the general knowledge, the memories refer to some past situations that have relevance in the story. For example, a memory can be referred to a past scene in which the character took part.

- Beliefs: The beliefs are a very subjective part of every character's cognition. They refer to facts about the world which the character considers as axioms, regardless of they are true. They can be part of the character's cultural or religious code, or simply originate in a particular misconception of the world.
- Dreams: The dreams represent the unconscious aspirations of the character. He/she may not be aware of them, but they can operate at a subconscious level and inspire his/her intentions.
- Fantasies: The fantasies are product of characters' imagination. They are beliefs or notions based on no solid foundation, a fact which the character is perfectly aware of. They represent aspirations that the character considers unreachable, but he/she enjoys thinking about them.
- Emotions: The emotions are related to the feelings of the character. They are usually influenced by the relationships that the character establishes with the others, and the evolution of them during the story.

Another relevant element of character's representation is the **function**. Although many approaches treat characters as entities subject to specific rules that interact in a simulated story world, there is another important line of thought in the treatment of characters: the functional view. In this perspective, pioneered by Aristotle [12] and followed by contemporary authors such as Propp [22], characters are subordinate to the narrative action. There are storytelling systems [10] that describe characters in terms of a structure based on their roles in the plot. Hence, the function tag refers to this approach and provides a way for defining the functional role of the character in the underlying structure of the story.

The **setting** is a combination of a set of physical —or virtual— locations in which the action of the story takes place, and the set of cultural and physical rules that govern the story world. The **locations** can be considered the scenario in which every scene that composes the plot takes place. So, as shown in the model, every scene links to its corresponding location.

6 Application

The model presented before aims at providing a reliable way of interchanging knowledge between different story generation systems, each of them using its own inner representation model.

The application of this model must necessarily rely on an architecture which combines different storytelling systems. For that purpose, it has been proposed a service-oriented architecture[4, 3] that combines three existing story generation systems: STellA (Story Telling Algorithm) [15], PropperWryter [10, 11], and Charade [18]. The involved systems will be adapted for packaging their basic functionalities as microservices that publish their capabilities as REST-based API [8]. This target architecture is summarized by Figure 2.

Every service will understand and generate messages for communicating with the others. The inner story generation processes of every system are unlikely to be interchangeable, but not the final product: the story. To this end, the proposed representation model is centred on the story, that is the common element among

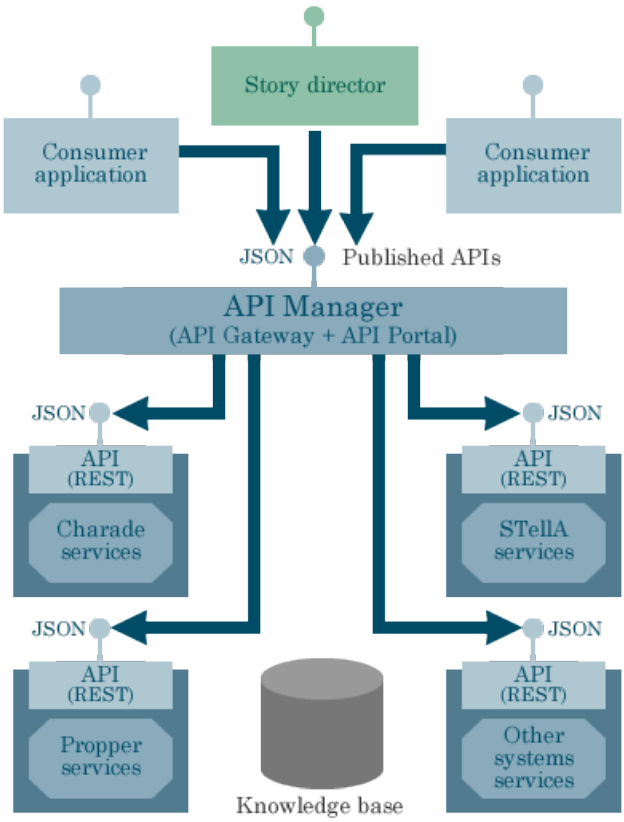


Fig. 2. Architecture of the proposed system.

the various systems. Therefore, by means of the common model it is possible to establish an iterative cycle of story generation, in which every service contributes to the story by generating a part of it. This way, the ability of PropperWryter is to develop the general scheme of the story, while STellA can provide a detailed simulation of the different scenes, and Charade describes the evolution of the relationships between the characters. All the required information for achieving this can be represented by means of the common model.

7 Conclusions

The model presented along this paper is based on the analysis of the knowledge required by existing storytelling systems. It tries to establish a baseline for make easier the interchange of knowledge between different story generation systems, and make possible the generation of richer and higher-quality stories by combining the capabilities of several systems.

In order to avoid to affect the internal story generation process of every participating system, the proposed model focuses on the product, that is the story. This approach seems to be the most convenient for keeping the common representation agnostic from the specifics of each system.

This knowledge representation model requires to be complemented by a architectural model which allow the development of a collaborative generation process involving various storytelling systems. Future work involves the development of a test field for this model, which allow to refine it and overcome its flaws. Currently, the authors are working in the target architecture, which involves the development of a storytelling ecosystem based on the referred systems —STellA, PropperWryter and Charade— as a benchmark for applying the common representation model.

Some of the essential points that will need to be evaluated are those related to knowledge consistency between systems, the possible degradation of the semantics, and the loss of relevant information during the communication between the participating systems. We considered that the future work on the proposed storytelling ecosystem will allow a complete study of these concerns, and also the improvement of the model to guarantee that the final result covers all the specified objectives.

Acknowledgements

This paper has been partially funded by the project IDiLyCo: Digital Inclusion, Language and Communication, Grant. No. TIN2015-66655-R (MINECO/FEDER).

References

1. Barthes, R.: S/Z: an essay. Siglo XXI (1980)
2. Chatman, S.B.: Story and discourse: Narrative structure in fiction and film. Cornell University Press (1980)
3. Concepción, E., Gervás, P., Méndez, G.: An api-based approach to co-creation in automatic storytelling. In: 6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017 (2017)
4. Concepción, E., Gervás, P., Méndez, G.: A microservice-based architecture for story generation. In: Microservices 2017 (2017)
5. Concepción, E., Gervás, P., Méndez, G., León, C.: Using cnl for knowledge elicitation and exchange across story generation systems. In: International Workshop on Controlled Natural Language. pp. 81–91. Springer (2016)
6. Davis, N., Hsiao, C.P., Popova, Y., Magerko, B.: An enactive model of creativity for computational collaboration and co-creation. In: Creativity in the Digital Age, pp. 109–133. Springer (2015)
7. Dehn, N.: Story generation after tale-spin. In: IJCAI. vol. 81, pp. 16–18 (1981)
8. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California, Irvine (2000)
9. Gervás, P.: Story generator algorithms. In: The Living Handbook of Narratology. Hamburg University Press (2012), http://hup.sub.uni-hamburg.de/lhn/index.php

10. Gervás, P.: Propp's morphology of the folk tale as a grammar for generation. In: OASIcs-OpenAccess Series in Informatics. vol. 32. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2013)
11. Gervás, P.: Reviewing propp's story generation procedure in the light of computational creativity. In: AISB Symposium on Computational Creativity, AISB-2014, April 1-4 2014. Goldsmiths, London, UK (04/2014 2014)
12. Halliwell, S., et al.: The Poetics of Aristotle: translation and commentary. UNC Press Books (1987)
13. Harrell, D.F.: Walking blues changes undersea: Imaginative narrative in interactive poetry generation with the griot system. In: AAAI 2006 Workshop in Computational Aesthetics: Artificial Intelligence Approaches to Happiness and Beauty. pp. 61–69 (2006)
14. Lebowitz, M.: Storytelling and generalization. In: Seventh Annual Conference of the Cognitive Science Society. pp. 100–109 (1985)
15. León, C., Gervás, P.: Creativity in story generation from the ground up: Non-deterministic simulation driven by narrative. In: 5th International Conference on Computational Creativity, ICCC (2014)
16. Margolin, U., Hühn, P., Meister, J.C., Pier, J., Schmid, W.: The living handbook of narratology (2013)
17. Meehan, J.R.: Tale-spin, an interactive program that writes stories. In: In Proceedings of the Fifth International Joint Conference on Artificial Intelligence. pp. 91–98 (1977)
18. Méndez, G., Gervás, P., León, C.: A model of character affinity for agent-based story generation. In: 9th International Conference on Knowledge, Information and Creativity Support Systems, Limassol, Cyprus. vol. 11, p. 2014 (2014)
19. Méndez, G., Gervás, P., León, C.: On the use of character affinities for story plot generation. In: Knowledge, Information and Creativity Support Systems, pp. 211–225. Springer (2016)
20. Montfort, N., Pérez, R., Harrell, D.F., Campana, A.: Slant: A blackboard system to generate plot, figuration, and narrative discourse aspects of stories. In: Proceedings of the fourth international conference on computational creativity. pp. 168–175 (2013)
21. Perez y Perez, R.: MEXICA: A Computer Model of Creativity in Writing. Ph.D. thesis, The University of Sussex (1999)
22. Propp, V.: Morphology of the folk tale. 1928 (1968)
23. Riedl, M.O., Young, R.M.: Narrative planning: balancing plot and character. Journal of Artificial Intelligence Research 39(1), 217–268 (2010)
24. Riedl, M.O.: Narrative generation: balancing plot and character. North Carolina State University (2004)
25. Schank, R.C., Abelson, R.P.: Scripts, plans, and knowledge. In: Proceedings of the 4th International Joint Conference on Artificial Intelligence - Volume 1. pp. 151–157. IJCAI'75, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1975), http://dl.acm.org/citation.cfm?id=1624626.1624649
26. Si, M., Marsella, S.C., Pynadath, D.V.: Thespian: Modeling socially normative behavior in a decision-theoretic framework. In: Intelligent Virtual Agents. pp. 369–382. Springer (2006)
27. Tolstoy, L.: War and peace. Courier Dover Publications (2017)
28. Veale, T.: Creativity as a web service: A vision of human and computer creativity in the web era. In: AAAI Spring Symposium: Creativity and (Early) Cognitive Development (2013)

Capítulo 4

Arquitectura

"Un edificio tiene dos vidas. La que imagina su creador y la vida que tiene. Y no siempre son iguales."
— Rem Koolhaas

4.1. Resumen de la arquitectura de referencia

El diseño de la arquitectura de referencia se ha realizado con vistas a disponer de una solución modular y distribuida que permita generar historias con múltiples tramas. Para ello, se ha tenido en cuenta la necesidad de identificar primero las etapas o dominios del proceso de generación, contando también con la necesaria cohesión interna de cada uno de los componentes extraídos. Todo este trabajo se ha concretado en Afanasyev, un modelo para la construcción de sistemas de generación de historias (Concepción et al., 2017a,b, 2018a). Afanasyev ofrece un modelo conceptual tanto a nivel de arquitectura como de representación del conocimiento. En el plano arquitectónico, plantea un marco de referencia para la construcción de sistemas generadores basado en componentes distribuidos. Y el modelo de representación del conocimiento permite el intercambio estructurado entre dichos componentes. Todo este trabajo se ha concretado en Afanasyev, que define una arquitectura de referencia para la construcción de sistemas de generación de historias (Concepción et al., 2017a,b, 2018a).

4.1.1. Estilo arquitectónico y características de la arquitectura

En líneas generales, se puede decir que la arquitectura de referencia pertenece a la familia de los estilos arquitectónicos basados en servicios, es decir, las arquitecturas orientadas a servicios (*Service-Oriented Architecture*, SOA) (Erl, 2004; Papazoglou, 2003), y las arquitecturas de microservicios (Newman, 2015; Wolff, 2016; Concepción et al., 2017c). Afanasyev define a nivel arquitectónico una colección de componentes orquestados por un servicio de alto nivel. El ecosistema global puede considerarse un pequeño ecosistema de microservicios que exponen una API, pudiendo considerarse como una *API Economy*, tal como la definen Gat y Succi (2013).

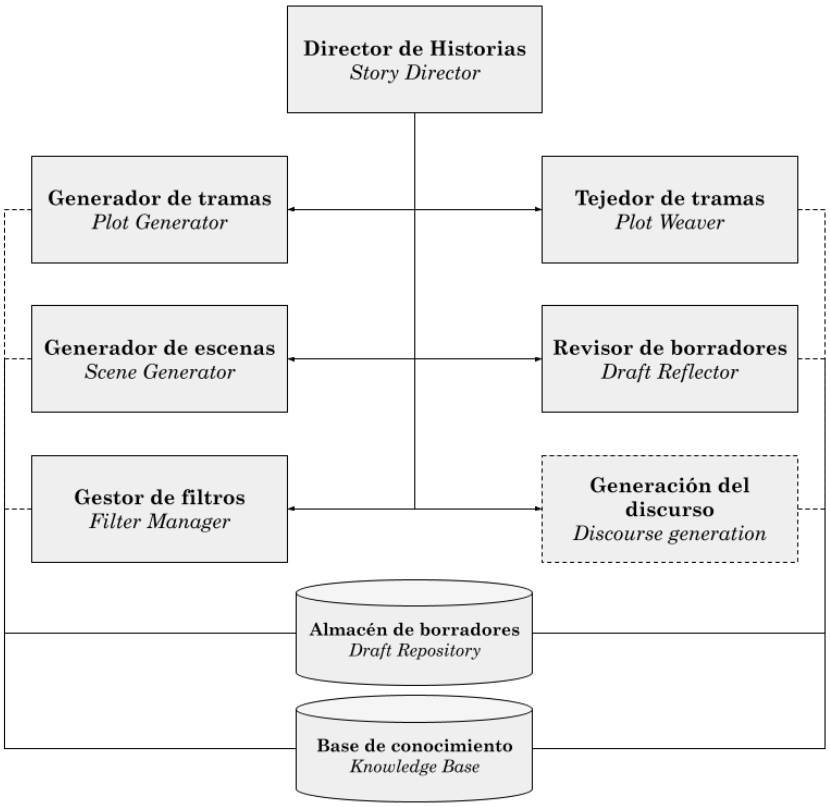


Figura 4.1: Arquitectura de referencia de Afanasyev.

Cada servicio expone sus capacidades mediante API REST (Fielding, 2000), que entiende y genera mensajes JSON (Crockford, 2006). Debido a que la lógica interna de cualquier microservicio es reemplazable mientras este preserve la interfaz que expone hacia el exterior, la arquitectura resultante ofrece una gran flexibilidad para cambiar el comportamiento interno de cada uno de sus componentes. Esta es la razón por la que Afanasyev incluye la definición de las interfaces REST comunes proporcionadas por los servicios y deja a cada servicio particular los detalles de la implementación. Este enfoque introduce varias ventajas. En primer lugar, toda la arquitectura está muy desacoplada. Esto significa que cada servicio se implementa y despliega por separado, y puede evolucionar independientemente de los demás. Otra ventaja de este modelo es que puede ampliarse en el futuro, añadiendo nuevos componentes al ecosistema sin afectar a los demás. Y por último, una característica muy importante, la facilidad para integrar nuevas heurísticas en la implementación de cada una de las etapas del proceso de generación de historias. Añadir un nuevo sistema de narración al ecosistema simplemente implica implementar al menos uno de los componentes de la interfaz y registrarlo para que opere junto al resto durante el proceso de generación.

Los componentes arquitectónicos de Afanasyev son los siguientes:

- Director de historias (*Story Director*)

- Generador de tramas (*Plot Generator*)
- Tejedor de tramas (*Plot Weaver*)
- Generador de escenas (*Scene Generator*)
- Gestor de filtros (*Filter Manager*)
- Revisor de borradores (*Draft Reflector*)
- Servicios de generación del discurso (*Discourse Planner, Sentence Planner, Linguistic Realization*)

La persistencia en Afanasyev está compuesta por dos almacenes: el Almacén de Borradores (*Draft Repository*) y la Base de Conocimientos (*Knowledge Base*). El Almacén de Borradores es una base de datos que almacena los borradores en curso y las historias finales. Actualmente, este componente se ha implementado utilizando una base de datos NoSQL (Han et al., 2011), MongoDB (MongoDB, 2017)). La base de conocimiento también es una base de datos NoSQL, pero en este caso concreto, es una base de datos orientada a grafos; en concreto, una Neo4j (Vukotic et al., 2014). Conserva todo el conocimiento compartido relacionado con conceptos, relaciones entre conceptos, reglas, etc. La figura 4.1 presenta de forma visual la organización de dichos componentes.

4.1.2. Modelo de operación de la arquitectura

Desde cierto punto de vista operativo, el funcionamiento de la arquitectura de Afanasyev puede evocar la idea de *Rayuela*, una novela de Cortázar (1963), cuyos capítulos pueden leerse en distinto orden, dando lugar a un buen número de interpretaciones válidas de la trama resultante. En este caso, la arquitectura proporciona la estructura y la función, que debe ser cubierta por los diferentes componentes que implementan cada API. Esto permite reemplazar componentes sin afectar al resto, o reconstruir un generador completo de acuerdo con la arquitectura de referencia (Concepción et al., 2017a). El conjunto está planteado para que opere de forma iterativa. En primer lugar, genera un borrador que será completado por los distintos servicios existentes en la arquitectura. El Director de Historias actúa como componente central, orquestando las peticiones a los distintos microservicios. El flujo que implementa es el que sigue:

- Generación de trama(s): El primer paso en la generación de una historia en nuestro modelo pasa por la generación de la trama. Este paso puede realizarse varias veces, dependiendo de si la historia que se desea generar consta de una o de varias tramas entretejidas. Como se verá en detalle más adelante, el modelo de operación soporta la creación de historias mono o multi-trama. En este caso, la generación de la trama se delega en el componente Generador de Trama (*Plot Generator*). El Director de Historias invocará al Generador de Trama tantas veces como tramas deban combinarse en la historia. Para poder realizar este paso, el Director de Historias necesita datos básicos del trasfondo de la historia

y de los personajes. Esto incluye la lista de personajes, básicamente una lista de parejas *(Nombre del personaje, Nombre del rol en la trama)*; una referencia al entorno o ambientación en la que tendrá lugar (de entre una lista de posibles ambientaciones); y, opcionalmente, una referencia a una plantilla base para la trama. En el caso en que se desee generar una historia con múltiples tramas, el Director de Historias verificará que la ambientación sea la misma en todas las llamadas, pues de lo contrario se podrían generar incoherencias. Los resultados generados en esta fase servirán para crear entradas en el Almacén de Borradores (*Draft Repository*).

- Entretejido de tramas: Este paso sólo se ejecutará si se está generando una historia con múltiples tramas. En caso afirmativo, se habrán creado varias entradas en el Almacén de Borradores que será necesario entretejer para obtener la historia con una trama múltiple. Esta labor será realizada por el Tejedor de Tramas (*Plot Weaver*). El Director de Historias invocará a dicho componente, pasándole las referencias del Almacén de Borradores con las historias cuyas tramas se desean entretejer. El resultado será una nueva historia cuya trama será la combinación de las tramas de las historias de entrada y cuyo conjunto de personajes será la unión de los diferentes conjuntos de personajes de las historias de entrada. Más adelante se describe con detalle cómo tiene lugar el proceso de reestructuración de historias en caso de entretejido de tramas.

- Generación de escenas: Como ya se ha definido previamente, la trama de una historia puede verse como el esqueleto que vertebra la historia. Para completar dicha metáfora, el proceso de generación de escenas consiste en agregar carne y músculo a dicho esqueleto. Como ya se ha detallado, la estructura de la trama en Afanasyev es una secuencia de escenas (Concepción et al., 2018a). El Director de Historias realizará tantas llamadas a este componente como escenas haya en la trama. En este caso, cada iteración dará en una actualización del borrador en el Almacén de Borradores. En esta fase, la única información de entrada necesaria es el propio borrador.

- Revisión del borrador: Esta etapa involucra actividades de diferente naturaleza orientadas a mejorar la calidad de los borradores. En Afanasyev se han definido varios componentes asociados a esta etapa: el Gestor de Filtros y el Revisor de Borradores.

- Generación del discurso: La última etapa, en la que la historia, ya completa y revisada, es convertida a texto en lenguaje natural, empleando el modelo clásico de generación de lenguaje natural (Gatt y Reiter, 2009). En esta fase es especialmente relevante el valor aportado por el modelo de representación del conocimiento, pues de la riqueza y calidad de la información que se haya acumulado en la historia, a lo largo del proceso de generación, dependerá la calidad del discurso generado.

Toda esta arquitectura de referencia se implementa en una primera versión de INES (*Interactive Narrative Emotional Storyteller*) (Concepción et al., 2018b), un

sistema generador de historias que implementa un subconjunto de los microservicios propuestos por Afanasyev para reescribir Charade (Méndez et al., 2016), un sistema preexistente.

4.1.3. Tratamiento de la arquitectura en los artículos

Los artículos recogidos en el presente capítulo siguen una progresión en cuanto a la definición detallada de los diferentes aspectos de la arquitectura de generación y culminan con la primera implementación del modelo de referencia de Afanasyev en INES.

- En el artículo "A microservice-based architecture for story generation" se plantea por primera vez la definición de una arquitectura basada en microservicios como solución a las necesidades de flexibilidad y modularidad que surgen de los objetivos de la investigación.
- En el siguiente artículo, "An API-based approach to co-creation in automatic storytelling", se enlaza el planteamiento arquitectónico con el modelo de representación del conocimiento y se definen con detalle las signaturas que deben cumplir las API asociadas a cada uno de los componentes.
- En "Afanasyev: A collaborative architectural model for automatic story generation" se completa la definición del marco completo y se establecen la operación, estructura y relación entre todas las partes de la arquitectura y la representación de una historia.
- En la línea evolutiva seguida desde los artículos anteriores, el artículo "INES: A reconstruction of the Charade storytelling system using the Afanasyev Framework" presenta una reescritura de Charade (Méndez et al., 2016), un sistema generador preexistente, aplicando la arquitectura de referencia de Afanasyev. El resultado de dicho trabajo es INES (Interactive Narrative Emotional Storyteller), un sistema generador basado en microservicios, que implementa un subconjunto de los definidos en el modelo de referencia. Este sistema permite poner a prueba las capacidades tanto de la arquitectura como del modelo de representación del conocimiento propuestos por Afanasyev.
- A la lista se añade, por último, "The long path to narrative generation", que compendia las técnicas aplicadas en generación automática y resume el enfoque de Afanasyev, entre otros.

4.2. Artículo: A microservice-based architecture for story generation

La principal novedad del presente artículo es la definición de una arquitectura distribuida y modular, basada en microservicios, para la implementación de un sistema generador. Como ya se ha expuesto en el Capítulo 2, son relativamente poco

comunes los sistemas generadores con una arquitectura diseñada con un enfoque modular. Desde el punto de vista arquitectónico, muchos sistemas de generación automática de historias se han diseñado tradicionalmente como sistemas monolíticos. Esta característica implica que una única aplicación concentra toda la funcionalidad y los activos necesarios. El concepto de "sistema monolítico" o "arquitectura monolítica", entendidos como un estilo arquitectónico, ha surgido principalmente tras la aparición de los Microservicios (Newman, 2015; Wolff, 2016). En una arquitectura monolítica, un único programa lo hace todo (Stephens, 2015). Muestra la interfaz de usuario, accede a los datos, procesa los pedidos de los clientes, imprime documentos y hace cualquier otra cosa que requiera hacer. Esta arquitectura presenta algunas limitaciones importantes. En particular, las piezas del sistema están estrechamente acopladas entre sí, por lo que no ofrece mucha flexibilidad. Esto afecta a varios atributos de calidad deseables de un sistema software (Kazman et al., 1999), como la mantenibilidad (*Maintainability*) y la capacidad de evolucionar (*Evolvability*), por mencionar algunos.

La principal ventaja de esta aproximación es que, a diferencia del diseño monolítico, que se traduce en que casi todos los sistemas generadores duplican una parte considerable de las funciones comunes de narración; en una arquitectura de microservicios cada componente se centra en una parte del proceso, y estos componentes podrían utilizarse por separado y evolucionar de forma independiente.

4.2.1. Cita completa

E. Concepción, Gervás, P., and Méndez, G. (2017). *A microservice-based architecture for story generation*, in Microservices, Odense, Denmark.

4.2.2. Resumen original

The present paper proposes an architectural model for knowledge interchange between story generation systems in the interest of enhancing the interoperability and fostering the co-creation process. The selected architectural approach is the microservices model, because it provides a very convenient way for structuring the functional responsibilities in the architecture, and promotes a distributed strategy for solving complex problems. The communication between the services is based on the well-known REST architectural model. This approach aims at simplifying the communication process by means of a easily achievable representation of the information

A microservice-based architecture for story generation

Eugenio Concepción[1], Pablo Gervás[12], and Gonzalo Méndez[12]

[1] Facultad de Informtica, Universidad Complutense de Madrid, Madrid, España
econcepc@ucm.es
[2] Instituto de Tecnología del Conocimiento, Universidad Complutense de Madrid
pgervas@sip.ucm.es gmendez@fdi.ucm.es

Abstract

The present paper proposes an architectural model for knowledge interchange between story generation systems in the interest of enhancing the interoperability and fostering the co-creation process. The selected architectural approach is the microservices model, because it provides a very convenient way for structuring the functional responsibilities in the architecture, and promotes a distributed strategy for solving complex problems. The communication between the services is based on the well-known REST architectural model. This approach aims at simplifying the communication process by means of a easily achievable representation of the information[1].

1 Introduction

Computational Creativity studies how to develop software that can take on some of the creative responsibility in arts and science projects. A story generator algorithm (SGA) refers to a computational procedure resulting in an artifact that can be considered a story [2].

Automatic story generation systems have been traditionally designed as monolithic systems from an architectural point of view. That means that a single application concentrated all the required functionality and assets. Obviously, this was a feasible solution for the earlier systems, which were built mainly for research purposes and implemented a limited-complexity functionality. As the story generation systems are becoming more complex, they are being designed in a much more modular way.

Despite there is not much literature on the subject, several efforts concerning collaborative story generation have been carried out. Slant [8] can be considered a significant example of storytelling systems interoperating for producing a enhanced outcome. It is an architecture for creative story generation that integrates different types of story generation systems. Slant also provides a convenient framework with a view to other systems to integrate with it.

Slant is the end result of an ambitious integration project that has involved many existing storytelling systems. From a technical point of view, Slant consists of a blackboard architecture that allows different storytelling systems or components to create stories collaboratively. The solution involves the integration of several different components from Mexica [9], GRIOT [5], and any other components specifically developed for this architecture.

2 A microservice-based architecture for story generation

The present paper proposes an architectural model for knowledge interchange between story generation systems in the interest of enhancing the interoperability and fostering the co-creation process. The co-creation model implies the involvement of several systems and humans working together in an iterative cycle of enhancement.

[1]This research has been supported by project IDiLyCo (MINECO/FEDER TIN2015-66655-R)

As mentioned in the previous section, many of the existing systems have been defined as monoliths, which make the collaboration between them a really complex challenge. This happens because almost every system duplicates a considerable part of main storytelling functions. For example, the generation of the story in natural language is a typical stage in every story generator. If every storytelling system break its architecture into finer-grain components, such as microservices, these components could be used separately. Also, every microservice would be autonomous enough to be independently evolved according new requirements, without affecting the rest of the architecture. But the most remarkable achievement of this approach would be the possibility of building hybrid coarse-grain services by composing the existing microservices. This new systems would take advantage of using the best-of-breed for building a collaborative story generation architecture.

This is precisely the aim of the architecture presented. It combines three existing story generation systems: STellA (Story Telling Algorithm) [6], PropperWryter [3, 4], and Charade [7]. The involved systems will be decomposed in its basic functionalities, that is, as microservices that will expose their capabilities as REST-based API. [1]. Every service will understand and generate JSON messages containing the required information in each case. Due to the fact that all these systems existed prior to the definition of the collaboration architecture, some parts can be considered as a legacy system that must be adapted to this new purpose. This is the reason why certain core capabilities of the systems will be reconstructed, and a new tier, specifically designed for publishing REST services, will be built for wrapping them. A high-level component, the composer, will implement the orchestration of the whole system, establishing the order in which every system would make its part.

References

[1] Roy Thomas Fielding. *Architectural styles and the design of network-based software architectures.* PhD thesis, University of California, Irvine, 2000.

[2] Pablo Gervás. Story generator algorithms. In *The Living Handbook of Narratology.* Hamburg University Press, 2012.

[3] Pablo Gervás. Propp's morphology of the folk tale as a grammar for generation. In *OASIcs-OpenAccess Series in Informatics*, volume 32. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

[4] Pablo Gervás. Reviewing propps story generation procedure in the light of computational creativity. *AISB 2014*, 2014.

[5] D Fox Harrell. Walking blues changes undersea: Imaginative narrative in interactive poetry generation with the griot system. In *AAAI 2006 Workshop in Computational Aesthetics: Artificial Intelligence Approaches to Happiness and Beauty*, pages 61–69, 2006.

[6] Carlos León and Pablo Gervás. Creativity in story generation from the ground up: non-deterministic simulation driven by narrative. In *5th International Conference on Computational Creativity, ICCC*, 2014.

[7] Gonzalo Méndez, Pablo Gervás, and Carlos León. A model of character affinity for agent-based story generation. In *9th International Conference on Knowledge, Information and Creativity Support Systems, Limassol, Cyprus*, volume 11, page 2014, 2014.

[8] Nick Montfort, Rafael Pérez, D Fox Harrell, and Andrew Campana. Slant: A blackboard system to generate plot, figuration, and narrative discourse aspects of stories. In *Proceedings of the fourth international conference on computational creativity*, pages 168–175, 2013.

[9] Rafael Perez y Perez. *MEXICA: A Computer Model of Creativity in Writing.* PhD thesis, The University of Sussex, 1999.

4.3. Artículo: An API-based approach to co-creation in automatic storytelling

Este artículo se centra en aplicar los conceptos apuntados en el anterior, relativos a la construcción de un sistema de generación como un ecosistema de microservicios, a la construcción de un sistema colaborativo con microservicios procedentes de tres sistemas generadores distintos: STellA (León y Gervás, 2014), Charade (Méndez et al., 2016) y PropperWriter (Gervás, 2016). Este artículo, además de constituir la contrapartida arquitectónica a *A common model for representing stories in automatic storytelling* (Concepción et al., 2017b), es también la continuación natural del trabajo de extracción de conocimiento en los mismos tres sistemas, reflejado en *Mining Knowledge in Storytelling Systems for Narrative Generation* (Concepción et al., 2016a).

4.3.1. Cita completa

E. Concepción, Gervás, P., and Méndez, G. (2017) *An API-based approach to co-creation in automatic storytelling*, in 6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017, Madrid, Spain.

4.3.2. Resumen original

The basic idea behind this paper is the development of a collaborative environment for generating stories. Hence, the authors put forward an architectural model for knowledge interchange between story generation systems, namely Propper, STella and Charade, in the interest of enhancing the interoperability and fostering the co-creation process. For this reason, this paper proposes an API Economy model based on the interchange of knowledge and services between story generation systems. The proposed architecture is based on an API-based microservices ecosystem connected according the REST architectural model. This approach aims at starting with a reduced set of services for extending it later with new capabilities.

An API-based Approach to Co-creation in Automatic Storytelling

Eugenio Concepción[1], Pablo Gervás[2], and Gonzalo Méndez[2]

[1] Facultad de Informática
[2] Instituto de Tecnología del Conocimiento
Universidad Complutense de Madrid
econcepc@ucm.es, pgervas@sip.ucm.es, gmendez@fdi.ucm.es

Abstract. The basic idea behind this paper is the development of a collaborative environment for generating stories. Hence, the authors put forward an architectural model for knowledge interchange between story generation systems, namely Propper, STella and Charade, in the interest of enhancing the interoperability and fostering the co-creation process. For this reason, this paper proposes an API Economy model based on the interchange of knowledge and services between story generation systems. The proposed architecture is based on an API-based microservices ecosystem connected according the REST architectural model. This approach aims at starting with a reduced set of services for extending it later with new capabilities.

Keywords: Computational creativity, Story generation systems, Software architecture, Service-Oriented Architecture

1 Introduction

Digital assets are increasingly becoming the most valuable resources that underlie much of the present economics. The digital artefacts are the key components in many organizations, whose businesses rely heavily on their ability to manage them. Making these key capabilities available by publishing them as APIs accelerates the innovation and provides uniform data and functionalities to internal and external actors. According to Willmott and Balas [28], an API Economy is defined as the emerging economic effects enabled by companies, governments, non-profits and individuals using APIs to provide direct programmable access to their systems and processes.

Automatic story generation is a part of a wider research area in Artificial Intelligence named Computational Creativity (CC), which aims to develop a creative behaviour in machines [26]. A story generator algorithm (SGA) refers to a computational procedure resulting in an artefact that can be considered a story [8]. A story generation system, also named storytelling system, can be defined as a computational system designed to tell stories [8].

From an architectural point of view, automatic story generation systems have been traditionally designed as monolithic systems. That means that a single

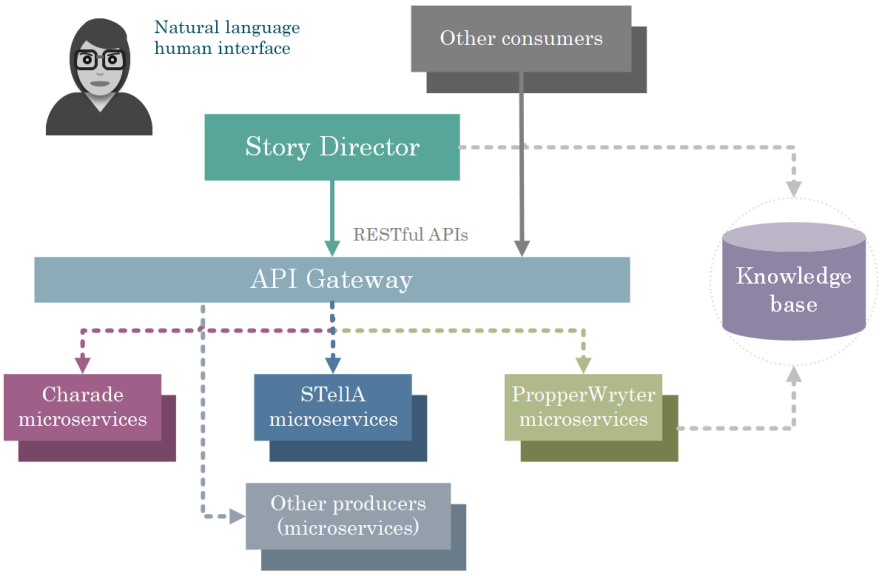


Fig. 1. An API-based architecture for storytelling.

application concentrated all the required functionality and assets. Obviously, this was a feasible solution for the earlier systems, which were built mainly for research purposes and implemented a limited-complexity functionality. As the story generation systems are becoming more complex, they are being designed in a much more modular way.

The big picture of the presented ecosystem relates to a service-oriented architecture [5, 20], and the microservices model [2]. This paradigm provides a convenient framework for organizing complex software systems. Applied to our particular research context, this approach, along with an API economy model, would allow the storytelling systems to create new functionalities and value. The resulting economy enables many new classes of applications with the potential to open new ways of hybridize algorithms, models and processes. This new ecosystem also entails the adoption of new roles, that is, API providers, API consumers, and the end user —as shown in Figure 1.

2 Related work

For the purpose of this paper, two research efforts need to be reviewed: a panorama of the architecture of the most relevant story generation systems – in order to understand how they operate and what type of architectural model they follow –, and existing approaches to combine story generation systems together – to consider what possible ways of combining them have been attempted.

Generally speaking, the architecture of a good part of the existing story generation systems usually fits with three main categories: those that are built over a planner [14, 4, 25, 23]; the systems developed by applying case-based reasoning (CBR) [25, 9]; and those builts over agent-based architectures [24, 15].

Despite there is not a vast amount of literature on the subject, several efforts concerning collaborative story generation have been carried out. Slant [17] can be considered a remarkable example of storytelling systems interoperating for producing an enhanced outcome. It is an architecture for creative story generation that integrates different types of story generation systems. It also provides a convenient framework with a view to other systems to integrate with it. Slant is the end result of an ambitious integration project that involves the integration of several different components: one based on Mexica [21], one based on GRIOT [11], and a new one specifically developed for the combined system. From a technical point of view, Slant consists of a blackboard architecture that allows different storytelling systems or components to create stories collaboratively. The goal is to allow the systems to influence each other for generating an enhanced result. The blackboard architecture for developing the story representation, and the Slant story XML format that is used, open up new possibilities for collaboration between creative literary systems, allowing models of creativity to be developed and added in different configurations.

In a wider context, still within the computational creativity area, it is also noteworthy the architecture proposed by Veale [26] for creative Web services. This model tried to combine both the academic and the industry needs in a solution for enhancing computational creativity systems. The architecture identified three types of services: *discovery services*, aimed at mining the knowledge contained in texts, and acquire emergent insights and novel perspectives on the expressed concepts; *composition services*, designed to suggest, elaborate, and comprehend conceptual metaphors, analogies, and blends, as well as services for accessing the large store of commonsense knowledge that these composition services will crucially rely upon; and *framing services*, which can package the conceptual conceit that underpins a creative act for an audience in a concise, easily appreciable, and memorable form, such as a linguistic metaphor, simile, joke, name, slogan, short story, poem, picture, piece of music, or a mixture of these forms. The proposed architecture is also accompanied by two specific Web services: Thesaurus Rex and Metaphor Magnet -as examples of creative functionality.

3 Scope

The idea behind the development of a collaborative environment for co-creating stories is pretty close to a practice referred by Veale [26] when he spoke of how organizations outsource their creative needs to external agencies. Such agencies act as option providers, in the sense that they create a universe of potential solutions, but leave others to make the final decision. In a co-creation scenario, several systems interact for creating a variety of feasible stories, but they require

the collaboration of one or more humans to evaluate the results and provide a feedback.

The involved systems, STellA [13], PropperWryter [9], and Charade [16], have been selected because they differ considerably from each other. The three systems focus on different aspects of storytelling. STellA is centred on causality, putting the stress on the causal order of events and actions. PropperWryter's thrust is the inner structure of the story. It works using the categories of characters functions defined by Propp [22]. Charade is basically oriented to the simulation aspect, giving as a result the evolution of the affinities between the characters. Thus, the combined operation of the three systems can be fruitful, especially if every component supplies the rest with its special features.

STellA can bring the basic causal structure of the story. This product can be refined by applying the functions defined in PropperWryter, giving a more cohesive plot. Charade can provide a more credible behaviour by incorporating the interaction between the characters of the story.

The choice for REST as the architectural style of the solution comes from the need of decoupling the distinctive features of each system from the communication architecture [12].

3.1 STellA

STellA (Story Telling Algorithm) [13] is a story generation system that controls and chooses states in a non-deterministically generated space of partial stories until it finds a satisfactory simulation of events that is rendered as a story.

STellA uses a custom representation for the knowledge it needs. It manages several different structures, including a matrix representation of the world in which characters live, and a set of rules for evaluating the range of results associated to the actions.

In STellA, the generation process involves an iterative creation of new states. Every simulation is modelled and implemented as a non-deterministic process in which every step can generate not only one but many others. This simulation requires the whole world domain to be explicitly represented as a simplistic view of a realistic environment. This approach provides a very detailed scenario that allows for a rich set of possibilities in generation. Each iteration generates a set of candidate versions of the current state, and then the process identifies the most likely ones by analysing their likelihood in terms of their plausibility and their narrative properties. This step is carried out by applying constraints and a generalized version of tension curves to drive story generation. These candidate partial stories are evaluated insofar they satisfy a given set of constraints and to what extent their tension curves fit with a set of target curves. The results of this process provide a criterion to decide if a partial story is promising and whether a story is finished.

The system requires an initial state and final conditions (which may also be a state to be reached). Basically, it generates from a starting point to a final condition. One of the most characteristic concepts managed by STellA is the *entropy*. The generator is able to generate many scenarios during the

reasoning process; and the more things are invented, the more entropy a story has. For example, if in the initial state there is a scenario expressed like *John loves Mary and they have a child*, and the child would be invented, it generates little entropy. Conversely, if the scenario expresses that *Mary has been abducted by the Martians*, it would be necessary to invent the *Martians*, who live on Mars, who want to take Mary (and why), and a few other things. That scenario would generate a lot of entropy.

Every state has entropy, and the state entropy is given by every generation cycle. The user determines how much entropy can be reached. The system outputs a very detailed sequence of snapshots of what happens at each moment. The result is a more or less narrative elaboration. Thus, the output is a list of states, in the same format as the input. Each state contains a timestamp. All the generated story, that is, all its states, are checked to see if they verify the established conditions.

3.2 PropperWryter

PropperWryter [9, 10] is a story generation system that creates Russian folktales according to Propp's generation rules [22]. These rules provide a very clear description of how the folktales morphology could be used for story generation. This approach has been previously used in other systems, like [27].

PropperWryter uses a set of abstractions for representing the essential concepts defined by Propp, especially the character function, and defines a procedure that first chooses a sequence of character functions to act as abstract narrative structure to drive the process, and then progressively selects instantiations of these character functions in terms of story actions to produce a conceptual representation of a valid story.

The generator can work in two forms: it can generate a story with no input, or it can take an input query (which can be a sequence of narrative tags or a sequence of actions), and generate a sequence of states. PropperWryter requires several work resources: a set of actions, a list of possible dependencies between actions, the mapping of each action to a high-level narrative label (Match, Return, Clash, Defeat, Prison, Release...), a list of possible dependencies between narrative labels (Departure-Return, Clash-Defeat, Prison-Release...), and the mapping of each variable that appears in an action to a narrative role (Hero, Villain, Victim...). The output of the system consists of a sequence of states, where each state is described by a set (not necessarily ordered) of predicates in which the characters are identified as variables. In general terms, this seeks to ensure that if the sequence includes an action that is an instance of a tag that has dependency on another tag, the sequence will also include an action that is an instance of this last tag. It is also intended that the assignment of narrative roles to each character appearing in the sequence (depending on the roles they play in the set of actions in which they appear) is consistent throughout the sequence.

3.3 Charade

The system developed by [15, 16] models the relationship between two characters using their mutual affinities, and applies it for generating stories. This system is an agent-based architecture developed using JADE. It consists of two types of agents: a Director Agent, that sets up the execution environment and creates the characters; and the Character Agents, one for each character of the story, whose interactions generate the story.

The main objective of the system were implementing an affinity model as decoupled as possible from the story domain, and testing it independently from other factors such as the environment in which the action takes place or the personality traits and emotional state of the characters. Due to this independence, it can be easily used to generate different kinds of stories.

The generator is based on a simulation of the characters' interaction. During the simulation, the characters perform actions between them, varying the affinity levels between them as a result. According to the affinity level, the characters can be a couple, friends, mutually indifferent, and enemies. Generation is independent of the domain; although, since it focuses on affinities, it works best in domains where this affinity makes sense. The simulation is not directed, so that it can not be considered to constitute a plot or a story by itself. The input includes a complete parametrization of possible actions, categorized according the type of relationship allowed for the characters, the simulated characters, and their relationships measured in terms of affinity. The output consists of a list of actions proposed by characters, and the response of their counterparts, that can accept or reject the proposals, with the variation of affinity between the characters involved. Despite no text is generated, it would be easy to use a template for generating a textual description.

4 Proposed solution

The proposed architecture aspires to articulate the operation of several automatic story generators in a way that allows the generation of higher quality stories by joining the capabilities of each system. The seed of this solution is the model composed by the three storytelling systems described before.

Every story generator provides a set of key operations in the form of services, so that each system can be considered as a module in the overall structure. The communications are based on the well-known REST architectural model [6]. This approach aims at simplifying the communication process by means of an easily achievable representation of the information. From a technical point of view, every involved system publishes their capabilities as REST-based services. Every service understands and generates JSON messages containing the required information in each case. Due to the fact that all these systems existed prior to the definition of the API ecosystem, everyone can be considered as a legacy system that must be adapted to this new purpose. This is the reason why the core capabilities or every system will be considered as the back-end, and a new

tier, specifically designed for publishing REST services, will be built for wrapping them.

4.1 Design considerations

Many of the existing story generation systems have been built in a such way that the collaboration between them is a really complex task. This happens because almost every system duplicates a considerable part of main storytelling functions. For example, the generation of the story in natural language is a typical stage in every story generator. If every storytelling system breaks its architecture into finer-grain components, such as microservices [29][18], these components could be used separately. Also, every microservice would be autonomous enough to be independently evolved according to new requirements, without affecting the rest of the architecture. But the most remarkable achievement of this approach would be the possibility of building hybrid coarse-grain services by composing the existing microservices. This new system would take advantage of using the best-of-breed for building a collaborative story generation architecture.

One of the key points of the architectural model is to ensure semantic interoperability. To develop a formalism for knowledge sharing in a collaborative architecture would make no sense if it is not possible to have an understanding by all parties of the information being exchanged. In this respect, it seems necessary a component for orchestrating the different microservices, and a repository to keep the shared knowledge that can be consulted by any microservice every time it has to interpret a request.

This model of knowledge base has been designed as centralized for two reasons: on the one hand, the concept-based framework applied must be necessarily shared between the various story generators, and on the other hand, it is necessary to avoid that the messages exchanged become too verbose. This last requirement is technical. If we understand that a microservice-based architecture is deployed in a distributed environment, this means that communications rely on the network, at all costs. If we assume that the communication model is truly REST, there is no state. This means that for each request it is necessary to send all the data that the server requires to be able to perform its work. This has the effect of sending the complete generated story in every request. In other words, the entire knowledge base required by the story is appended to the request data, making the communications inevitably become inoperative after a few requests between systems.

Another aspect to consider is that it is not possible to delegate to each system the definition of the entities or concepts that it handles. Take, for example, an action as simple as eating. In the case of Charade, this concept refers to a couple lunch, and it is an atomic action. Instead, in STellA this is a composite action, which refers to the physical act of eating and involves several steps such as bringing food to the mouth, chewing and swallowing. Clearly, a human being senses that the same concept is not being talked about, but a computer system needs to have precise definitions so that it knows what the system is referring to.

It is therefore necessary for the definition to become universal, and for systems to know what they are referring to by taking existing concepts to operate.

4.2 Methodology

The involved systems will be decomposed in its basic functionalities, that is, as microservices that will expose their capabilities as REST-based API [6]. Every service will understand and generate JSON messages containing the required information in each case. Due to the fact that all these systems existed prior to the definition of the collaboration architecture, some parts can be considered as a legacy system that must be adapted to this new purpose. This is the reason why certain core capabilities of the systems will be reconstructed, and a new tier, specifically designed for publishing REST services, will be built for wrapping them. A high-level component, namely the story director, will implement the orchestration of the whole system, establishing the order in which every system would make its part. For achieving a full syntactic and semantic interoperability, the exchanged messages between the different components will be based on a common knowledge representation model [1].

The steps for achieving the establishment of a well-grounded API Economy are widely discussed by specialized literature [7] [19].

Olson [19], suggested that organizations should treat their API as products it must nurture. In this regard, she proposed a sequence of steps for achieving this [19]. Notable among them are the importance of understanding the value chain, and the establishment of goals for every API strategy.

4.3 System design

The general model of joint operation of the three systems is based on the use of the key capabilities of each of them. The Figure 2 depicts the whole architecture and its components. Thus, the role of PropperWryter is to develop the main scheme of the plot, while STellA is responsible for simulating the development of the different low-level scenes, and Charade establishes the evolution of the relationships between the characters.

The role of the story director entails the orchestration of the whole system, establishing the order in which every system would make its part. This is the central component that will need to preserve the collective knowledge by means of the common knowledge base. Concerning this point, the need for a common representation arises. As stated above, every system focuses in a different aspect of story generation, so are its knowledge representation. Every published service must be considered to provide system-specific knowledge structure, so the adaptation step must be performed in the composer module.

The story director is also related to the maintenance of consistency in the story that is being generated. As already mentioned, Charade simulates the evolution of relationships (couple, friends or enemies). Let us suppose a story in which, in one of the scenes of the plot, the couple has a romantic dinner and the end result is that their love affinity increases. For Charade, a romantic

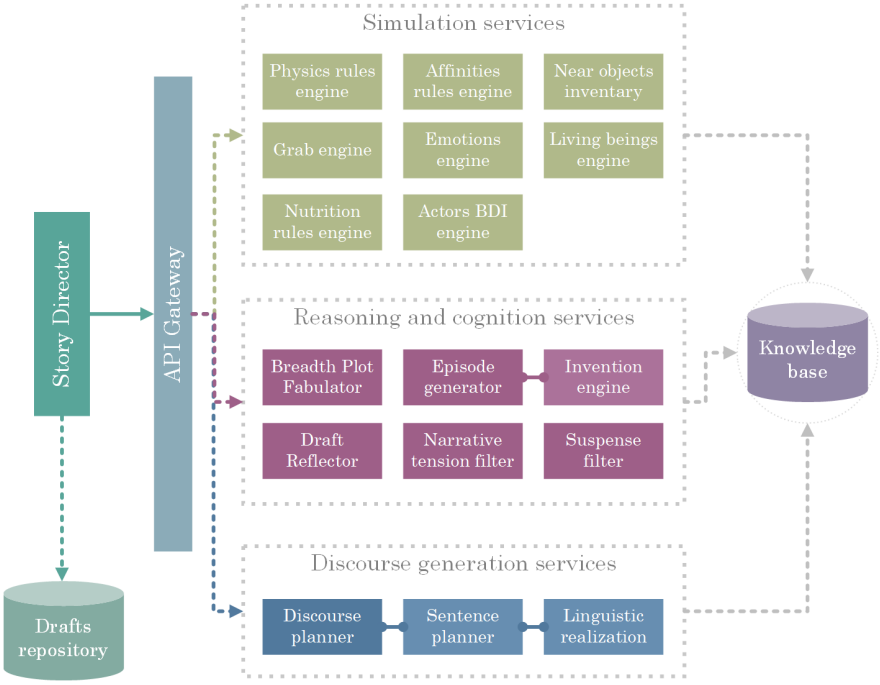


Fig. 2. Architecture of the proposed system.

dinner is an atomic action, it does not go into the detail of how it evolves. The story could then be passed to a service of STellA for developing in more detail the scene of the romantic dinner. During the simulation, STellA generates the actions performed by the characters, and it turns out that, at a given moment, the couple conversation becomes a discussion. This result would be clearly inconsistent with the final result calculated previously. At this point, the director's role becomes crucial. It must decide whether to discard the scene generated by STellA, if it changes the course of events in the relationship (as generated by Charade), or if it requests STellA to re-evaluate the situation so that after the discussion there is a reconciliation, and the dinner ends happily.

In a first approach, a REST-based interface is being defined for wrapping the original story generation systems. This step leads to the definition of a set of common concepts shared across the systems, so that the same entities are expressed in the same way. This step is essential for articulating a generation pipeline with all the systems. At this point, there are key concepts that must be managed by the director and clearly informed to the participants: state transitions, actions and results (understood as a causal chain), sequentiality...

The key component of the API Economy approach is the API Manager. This component provides the scaffolding for building the service-based API structure. Usually, an API Manager provides three essential features: a single entry point for

all the consumers requests (API Gateway), a central web-based tool for managing the various policies to be applied, and a marketplace for developers that allows them to easily find the APIs they need to consume (API Portal). All these component are normally completed by a centralized configuration service, and the elastic infrastructure for hosting the microservices. The roles and derived benefits from this infrastructure are multiple:

- Centralized configuration, a service that all applications use to specify and access their respective configuration information in a consistent way.
- Automatic deployment, a service that invokes and decommissions APIs and service implementations under administrator control.
- Single security enforcement point, usually provided by the API Gateway.
- Auditing and monitoring, provided by the API Gateway.

Another essential component in architecture is the repository of drafts (or stories). Both PropperWryter and STellA generate story trees from possible continuations. While the STellA model is less restrictive than the PropperWryter model, in both cases it is necessary to maintain a draft tree in progress. In order to avoid having an exchange of excessively large JSON messages, the idea is to reposition all the drafts, to recover them only when required. The formalism employed for representing these drafts is detailed in a specific paper [1].

Thus, the main services are the following:

- BreadthFabulator (PropperWryter)
- Reflector (PropperWryter)
- Simulation engines (STellA)
- EpisodeGenerationEngine (STellA)
- DiscourseGenerator (STellA)
- SuspenseFilter (STellA)
- NarrativeTensionFilter (STellA)
- AffinitiesEngine (Charade)
- StoryDirector

In the development of the plot, the actions of the characters and the events modify the global state of the narrative universe. In this sense, every action that takes place in the plot carries information related to the new state in which the universe remains.

The joint operation of the microservices ecosystem will be directed by the Story Director, who will act as an orchestrator of the generation process. It will request the APIs of the different services according to the generation process. This process will proceed iteratively, generating drafts that will be refined in each pass, until the established criteria for story completeness are met. The Reflector service will analyse the draft for ensuring the compliance of these criteria —as it originally did in PropperWryter.

In the case of STellA, it provides a detailed simulation for every scene generated by PropperWryter in the high-level plot. The service that provides this is the Episode Generation Engine, which receives as input a draft of the story,

which contains information about the characters, and what must happen in the scene at a high level. The service then generates a simulation to explore the universe of possible solutions. Unlike the original operation of STellA, which was unrestricted, in this case, there are restrictions to apply to the final state in which the scene must be found. This means that there will be solutions whose generation should be truncated by not reaching a state of the narrative universe compatible with the expected final state. The result of the simulation will be a new collection of drafts that will be persisted in the Drafts Repository. The director of the story, who is responsible for orchestrating the behaviour of the whole, will analyze the various drafts through by means of the two filtering microservices (Suspense and narrative tension). The objective is discarding the stuff that should not prosper in the next iteration. The different STellA simulation engines, as well as the knowledge base, will be used at convenience. This is also the case of the affinity engine of Charade, which will serve to calculate the evolution of the relationship of the characters as the development of the plot takes place. In this sense, it is important that a relationship be established between all the concepts that the three systems handle. For example, in the case of an action such as dining, which can be interpreted in different ways by each of the systems, especially in the case of STellA, which tends to a strong physical representation of the actions.

5 Conclusions and future work

The set out approach intends to establish a collaborative model that allows the free exchange of knowledge between the different storytelling systems in order to develop an iterative improvement process of literary creation. In addition to this objective, it promotes the development of a knowledge representation model for creating a common knowledge base that can be fed in the future with the outcomes of new storytelling systems, without the need to adapt locally their knowledge representation models.

The architecture exposed so far has several features to be developed in the next steps. The main pending task is related to the design and implementation of the composition service. It is necessary to develop, not only a technical model for aggregating services, but also a proper interface for interacting with the human participants in the process.

Currently, in every iteration, for every draft of the current population, all the possible continuations are generated and added to the population of the next iteration. On the generated population, a reflection process is applied (Reflector class), and drafts that are considered already finished are separated from the work population. This process continues until the work population is empty (all drafts are terminated) or a limit of iterations is reached (to guarantee completion). In the face of future work, the development of a service that helps to decide is pending. The process for deciding what is the most appropriate level of detail in each of the scenes is still pending. If we take as an example any novel, it can be seen that in each scene a different level of detail is handled —which greatly

influences the narrative rhythm, for example. Certain scenes are described at a high level, without going too deeply into the details, while other scenes related to very brief moments in time, are treated in detail, because they are very relevant in the narration. This component will become increasingly important as more and more systems are incorporated into the proposed ecosystem.

The most immediate roadmap focuses on developing and testing the described REST-based services. Once these services become available, the next step will involve the design of a formal representation for the persisted knowledge. On the basis of this knowledge, the composer could generate a human-readable output. As stated in [3], a suitable solution would be the use of a controlled natural language (CNL), which is naturally easier to understand by humans than formal languages, encouraging the co-creation cycle.

Once that all the participants have implemented and made available their services, the next step will be the development of the process for integrating them in a generation pipeline making use of the knowledge shared across them. It is still a matter of study how to apply certain local concepts, such as entropy, to the whole architecture for enhancing the global outcomes.

Acknowledgements

This paper has been partially funded by the project IDiLyCo: Digital Inclusion, Language and Communication, Grant. No. TIN2015-66655-R (MINECO/FEDER).

References

1. Concepción, E., Gervás, P., Méndez, G.: A common model for representing stories in automatic storytelling. In: 6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017 (2017)
2. Concepción, E., Gervás, P., Méndez, G.: A microservice-based architecture for story generation. In: Microservices 2017 (2017)
3. Concepción, E., Gervás, P., Méndez, G., León, C.: Using cnl for knowledge elicitation and exchange across story generation systems. In: International Workshop on Controlled Natural Language. pp. 81–91. Springer (2016)
4. Dehn, N.: Story generation after tale-spin. In: IJCAI. vol. 81, pp. 16–18 (1981)
5. Erl, T.: Service-oriented architecture: a field guide to integrating XML and web services. Prentice Hall PTR (2004)
6. Fielding, R.T.: Architectural styles and the design of network-based software architectures. Ph.D. thesis, University of California, Irvine (2000)
7. Gat, I., Succi, G.: A survey of the api economy. Cut. Consort (2013)
8. Gervás, P.: Story generator algorithms. In: The Living Handbook of Narratology. Hamburg University Press (2012), http://hup.sub.uni-hamburg.de/lhn/index.php
9. Gervás, P.: Propp's morphology of the folk tale as a grammar for generation. In: OASIcs-OpenAccess Series in Informatics. vol. 32. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2013)
10. Gervás, P.: Reviewing propp's story generation procedure in the light of computational creativity. In: AISB Symposium on Computational Creativity, AISB-2014, April 1-4 2014. Goldsmiths, London, UK (04/2014 2014)

11. Harrell, D.F.: Walking blues changes undersea: Imaginative narrative in interactive poetry generation with the griot system. In: AAAI 2006 Workshop in Computational Aesthetics: Artificial Intelligence Approaches to Happiness and Beauty. pp. 61–69 (2006)
12. Khare, R., Taylor, R.N.: Extending the representational state transfer (rest) architectural style for decentralized systems. In: Software Engineering, 2004. ICSE 2004. Proceedings. 26th International Conference on. pp. 428–437. IEEE (2004)
13. León, C., Gervás, P.: Creativity in story generation from the ground up: Non-deterministic simulation driven by narrative. In: 5th International Conference on Computational Creativity, ICCC (2014)
14. Meehan, J.R.: Tale-spin, an interactive program that writes stories. In: In Proceedings of the Fifth International Joint Conference on Artificial Intelligence. pp. 91–98 (1977)
15. Méndez, G., Gervás, P., León, C.: A model of character affinity for agent-based story generation. In: 9th International Conference on Knowledge, Information and Creativity Support Systems, Limassol, Cyprus. vol. 11, p. 2014 (2014)
16. Méndez, G., Gervás, P., León, C.: On the use of character affinities for story plot generation. In: Knowledge, Information and Creativity Support Systems, pp. 211–225. Springer (2016)
17. Montfort, N., Pérez, R., Harrell, D.F., Campana, A.: Slant: A blackboard system to generate plot, figuration, and narrative discourse aspects of stories. In: Proceedings of the fourth international conference on computational creativity. pp. 168–175 (2013)
18. Newman, S.: Building microservices: designing fine-grained systems. " O'Reilly Media, Inc." (2015)
19. Olson, L.: The open api economy: What is it and how do i capitalize on it? In: International Service Technology Symposium (2012)
20. Papazoglou, M.P.: Service-oriented computing: Concepts, characteristics and directions. In: Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on. pp. 3–12. IEEE (2003)
21. Perez y Perez, R.: MEXICA: A Computer Model of Creativity in Writing. Ph.D. thesis, The University of Sussex (1999)
22. Propp, V.: Morphology of the folk tale. 1928 (1968)
23. Riedl, M.O., Young, R.M.: Narrative planning: balancing plot and character. Journal of Artificial Intelligence Research 39(1), 217–268 (2010)
24. Si, M., Marsella, S.C., Pynadath, D.V.: Thespian: Modeling socially normative behavior in a decision-theoretic framework. In: Intelligent Virtual Agents. pp. 369–382. Springer (2006)
25. Turner, S.R.: Minstrel: A Computer Model of Creativity and Storytelling. Ph.D. thesis, University of California at Los Angeles, Los Angeles, CA, USA (1993), uMI Order no. GAX93-19933
26. Veale, T.: A service-oriented architecture for computational creativity. Journal of Computing Science and Engineering 7(3), 159–167 (2013)
27. Wama, T., Nakatsu, R.: Analysis and generation of japanese folktales based on vladimir propps methodology. In: New Frontiers for Entertainment Computing, pp. 129–137. Springer (2008)
28. Willmott, S., Balas, G.: Winning in the api economy. 3scale, octobre (2013)
29. Wolff, E.: Microservices: Flexible Software Architecture. Addison-Wesley Professional (2016)

4.4. Artículo: Afanasyev: A collaborative architectural model for automatic story generation

Este artículo presenta el modelo arquitectónico de referencia pensado para para asegurar la construcción de sistemas generadores modulares que faciliten la colaboración entre componentes autónomos, permitiendo incluso la integración de implementaciones procedentes de sistemas distintos. Esto se traduce en el diseño de interfaces bien definidas que permiten la aplicación de implementaciones intercambiables en un marco arquitectónico general con funciones claramente definidas para cada componente y un modelo común de representación del conocimiento.

4.4.1. Cita completa

E. Concepción, Gervás, P., and Méndez, G. (2018) *Afanasyev: A collaborative architectural model for automatic story generation*, in 5th AISB Symposium on Computational Creativity, University of Liverpool, UK.

4.4.2. Resumen original

The present article focuses on detailing the characteristics of Afanasyev, an architectural framework for the construction of story generation systems through replaceable services. The basic idea behind this approach is the development of a collaborative environment for generating stories. This entails the inclusion of a common representation model to allow the interoperation between different story generation systems as a base for a collaborative environment to run an enhanced process of literary creation. In addition to this objective, this model aims at the development of a story representation formalism for creating a common knowledge base that can be fed in the future with the outcomes of new storytelling systems, without the need to adapt it to every system-specific representation model.

Afanasyev: A collaborative architectural model for automatic story generation

Eugenio Concepción and **Pablo Gervás** and **Gonzalo Méndez**[1]

Abstract. The present article focuses on detailing the characteristics of Afanasyev, an architectural framework for the construction of story generation systems through replaceable services. The basic idea behind this approach is the development of a collaborative environment for generating stories. This entails the inclusion of a common representation model to allow the interoperation between different story generation systems as a base for a collaborative environment to run an enhanced process of literary creation. In addition to this objective, this model aims at the development of a story representation formalism for creating a common knowledge base that can be fed in the future with the outcomes of new storytelling systems, without the need to adapt it to every system-specific representation model.

1 INTRODUCTION

Automatic story generation is a long-standing research field in the area of Computational Creativity (CC), which pursues the development of creative behaviour in machines [42]. A story generator algorithm (SGA) refers to a computational procedure resulting in an artefact that can be considered a story [19]. In other words, a story generation system is a computational system designed to tell stories. So, the terms story generation system and storytelling system can be considered equivalent.

From an architectural point of view, many automatic story generation systems have been traditionally designed as monolithic systems. This feature entails that a single application concentrates all the required functionality and assets. While this was a feasible solution for the earlier systems, mainly designed for research purposes and a limited-complexity functionality, nowadays it seems quite difficult to host the ideally expectable storytelling capabilities with such model. So, as the story generation systems are becoming more complex, they are being designed in a much more modular way.

This paper introduces Afanasyev, a collaborative architectural model for automatic story generation which relates to a service-oriented architecture (SOA) [11, 36], and the microservices model [6]. The SOA paradigm provides a convenient framework for organizing complex software systems. In addition, the main contribution of the microservices architectural pattern to the service-based landscape is the development of highly distributed and decoupled applications. The application of this approach to the context of automatic story generation, along with the concepts taken from the API economy model [18], would allow the storytelling systems to create new functionalities and value.

This document is structured in four main blocks: a general review of the existing storytelling systems, with a special emphasis on collaborative story generation; a summarized statement of the problem; a detailed description of the proposed solution; and a final part focused on discussing some specific aspects of the solution and the conclusions.

[1] Universidad Complutense de Madrid, Spain, email: econcepc@ucm.es, pgervas@sip.ucm.es, gmendez@fdi.ucm.es

2 BACKGROUND

The first story generation systems date back to the 1970s. The Automatic Novel Writer [26] is considered the first storytelling system. It generated murder stories in a weekend party setting. Its capabilities were quite limited, so the generated stories had an identical structure and the only variation came from the characters roles.

TALE-SPIN [32] was another of the earlier story generators. It was a planning solver system that wrote up a story narrating the steps performed by the characters for achieving their goals. TALE-SPIN generated stories about the inhabitants of a forest taking a collection of characters with their corresponding objectives as inputs. TALE-SPIN found a solution for those characters goals, and wrote up a story narrating the steps performed for achieving those goals.

Author [10] was the first story generator to include the author's goals as a part of the story generation process. Dehn considered that stories were mainly the result of a plot conceived in author's mind. In such a way, Author intended to emulate the mind of a writer. Conceptually it was a planner but, unlike TALE-SPIN, it used the planning to fulfill authorial goals instead of character goals.

Universe [28] was designed for generating the scripts of a TV soap opera episodes in which a large cast of characters played out multiple, simultaneous, overlapping stories that could continue indefinitely, without a closed end. Universe gave a special importance to the creation of characters, in contrast with Dehn's approach. It used complex data structures for modelling characters, using as input both predefined stereotypes and user-provided characterization.

Mexica [37] was developed as a computer model whose purpose was studying the creative process. It generated short stories about the early inhabitants of Mexico. Mexica was a pioneer in that it took into account emotional links and tensions between the characters as a means for driving and evaluating ongoing stories.

Fabulist [38] is a complete architecture for automatic story generation and presentation. Fabulist combines an author-centric approach together with a representation of characters intentionality, and an open-world planning for maximizing the quality of the stories.

Curveship [34] was a system for interactive fiction in which the user controls the main character of a story by introducing simple descriptions of what it should do, and the system generates descriptions of the outcomes of the character's actions. Curveship's storytelling approach differs from other story generation systems in the sense that it tells the story from different perspectives, without modifying

the plot. For example, it makes use of a wide variety of techniques such as flashback, flash-forwards, interleaving of events from two different time periods, telling events back to front.

Regardless of whether the construction of the story plots relied on grammars [26], planning [32, 10, 28], or case-based reasoning [41, 21], a good part of the mentioned storytelling systems fitted the monolithic model. In addition to this approach, simulation-based systems [38, 34] were built mainly as distributed architectures. None of the aforementioned generators combined capabilities from other systems, nor considered the collaboration with others.

Slant [35] can be considered a remarkable example of storytelling systems working collaboratively for producing an enhanced outcome. It is an architecture for creative story generation that integrates several components from different systems: Mexica [37], Curveship [34] and Griot [24]. The latter is a collection of Computational Creativity related systems. The core of Griot is Alloy, a component which makes what its authors name "blending"[22]. Conceptual blending is an idea that comes from cognitive linguistics. It is a model of creative thinking in which two concepts can be integrated to form a new one. Namely, the thrust of this approach is the integration of different concepts in order to produce some creative results —for example, metaphors.

In a wider context, still within the computational creativity area, it is noteworthy the architecture proposed by Veale [42] for creative Web services. In an effort to accomplish both the academic and the industry needs, he proposes a solution for enhancing computational creativity systems by introducing an architectural model which categorizes the services according to their function in the application structure.

After the prior analysis of a representative subset of the existing storytelling systems, it seems quite clear that every system has been designed according to certain operational expectations that they are able to accomplish, but they difficultly can produce stories beyond their predefined target model. Hence, it is quite uncommon to find a single story generation system producing stories that combine different narrative rhythms or that deal with diverse motifs in the thematic aspect.

3 STATEMENT OF THE PROBLEM

What makes a story captivating? The basic elements of a story have been largely analysed by classic Narratology [3, 2, 31]. The plot is an essential element in a story, but so are the characters depiction, the narrative discourse, the rhythm, the emotional arc and many others. All these elements produce an effect in the people watching a play or a film, reading a novel or listening to a narrator. The wise arrangement of all these components, adapting the length of each scene to the most convenient one, varying the speech and description passages, choosing the right timing for the key events and remaining faithful to the theme, help to create movement, tension and emotional value in the development of the story.

Despite the efforts made in the field of automatic story generation, the stories written by humans are considerably more complex than those generated by computational systems. Consider as an example any classic novel: they contain a main plot, several subplots, every chapter can be focused on a different theme, there are changes in the rhythm of the narration, there are passages that focus on a particular character and ignore the rest, and many other features that help to keep the readers attention in the narration. The existing storytelling systems are capable of creating a single-themed story, with a single narrative structure and a specific rhythm.

Coupled with the intrinsic limitations of the generation model, the monolithic architecture of many existing systems introduces an additional limiting factor.

Considering the collaboration between different storytelling systems as a simple way of generating more natural stories, it seems appropriate that a solution could involve using different systems, generating different types of content according to their capabilities. Due to the fact that a monolithic design hinders the collaboration with other systems, this paper considers the use of several systems working collaboratively for achieving the generation of richer and more complex stories by providing a service-based framework for automatic storytelling. This approach would allow to combine different services from different story generation models –or systems, so the outcome would be closer to the diversity of narrative resources that characterize the stories created by humans.

4 PROPOSED SOLUTION

Many of the existing systems have been designed as monoliths, which make the collaboration between them a really complex challenge. This happens because almost every system duplicates a considerable part of the common storytelling functions. If every storytelling system broke its architecture into finer-grain components, such as microservices, these components could be used separately and evolve independently.

The basic idea of the proposed solution can be seen as one of those toddler toys in which they have to classify different pieces by matching the shapes and drop every block through the sorter. In this case, the model supports the use of different types of automatic storytelling services, as long as they can implement every required interface.

Afanasyev is basically a collection of microservices orchestrated by a high-level service. The overall ecosystem can be considered a small storytelling API Economy [18]. Each service exposes their capabilities as REST-based API [13] and it understands and generates JSON messages. Due to the fact that the inner logic of any microservice can come from a different storytelling system, its interface must be adapted to this new purpose. This is the reason why Afanasyev includes the definition of the common REST interfaces provided by the services and leaves to every particular system the details of the implementation. This approach introduces several benefits. First of all, the whole architecture is highly decoupled. This means that every service is implemented and deployed separately, and it can evolve independently from the others. Another benefit of this model is that it can be extended in the future, by adding new microservices to the ecosystem without affecting the others. And finally, a very important feature, the ease of integrating a new system. To add a new storytelling system to the ecosystem, simply entails to implement at least one of the microservices interface, and registering it in order to be considered by the Story Director during the generation process.

From a certain point of view, the operation of Afanasyev may evoke the idea behind *Hopscotch*, a novel by Cortázar[9], whose chapters can be read in different order, giving rise to a good number of differing valid interpretations of the resulting plot. In this case, the architecture provides the structure and function, which must be covered by the different microservices that implement each API. This allows to use parts coming from different generating systems in a combined way, or to reconstruct a complete generator according to the architecture provided by the framework. An early approach to this model was proposed as part of a wider API-based collaborative environment [4].

The development of Afanasyev entails two main tasks: the defini-

tion of a shared knowledge representation model and the design of a microservice-based architectural environment. Both are addressed in the following sections.

4.1 Common knowledge representation model

In order to allow the combined operation, the microservices of the framework require a common representation model for stories. The knowledge required to generate stories depends heavily on a number of factors. One of these key factors is the system architecture. The components that participate in the generation process condition the structure of the knowledge. For example, in the case of storytelling systems built over planners, it is necessary to keep knowledge concerning states, preconditions, actions, effects of the actions, etc. Grammar-based story generators require a complete representation of the applicable rules for creating their stories. Simulation-based storytelling requires a detailed typification of the characters and their relationships. On the other hand, there is a common element for every storytelling system that can be interchanged: the story, which is the end product of the generation process.

The proposed representation model [5] focuses on the knowledge that is directly related to the story, instead of that related to the generation process, which would be hard to export between different systems. This model is strongly influenced by the components of narrative identified in the classic Narratology [3, 2, 31]. These concepts and structure are enhanced by various storytelling-related computational concerns.

The resulting representation model is summarized in Figure 1.

The model has been designed as a hierarchical structure, in which the root concept is the **story**. Most of the leafs of this tree-like structure are asserts representing a piece of knowledge. These asserts are expressed by means of sentences in a Controlled Natural Language (CNL) [39]. The use of a CNL for representing knowledge in storytelling systems has been proposed by the authors in earlier papers [7, 8]. The main advantage of using a CNL is that the concepts referred in the asserts can be expressed by domain experts in the knowledge base and then they can be translated to the variety of formal representations used by the various services. This feature allows the definition of rules in a system-agnostic language, useful not only for expressing the different concepts involved in the story, but also for exchanging these knowledge resources across the different storytelling services.

A story represents what both intuitively and narratologically can be considered a story, that is, a narration of the actions performed by the characters and the events happening in a setting. A story is composed by two main elements: the plot and the space.

The **plot** is represented as a sequence of scenes. A **scene** is conceptually related to the division of a play, that represents a single episode inside the plot. It is clearly conditioned by the time division, which means that it is a sequence of events that happen during a time frame. From a spatial point of view, it is also constrained to take place in a single spatial frame —considering the spatial frame definition mentioned before. So, the scene is composed by a sequence of **events**, that can be actions or happenings. An **action** is an act performed by one or more characters in the story, generating consequences. The resulting consequences of every action are expressed as a modification in the global state of the space —considering it as the whole setting and the existents. A **happening** is an event that happens in the plot, as an accident or as a consequence of a prior action or happening. A happening can be natural —it rains— or artificial —a car accident. Regardless of the type of event, both are characterized by their impact in the story world. This is represented as a pair of states: the previous state and the later state. Each state is represented by a set of asserts, expressed in a CNL.

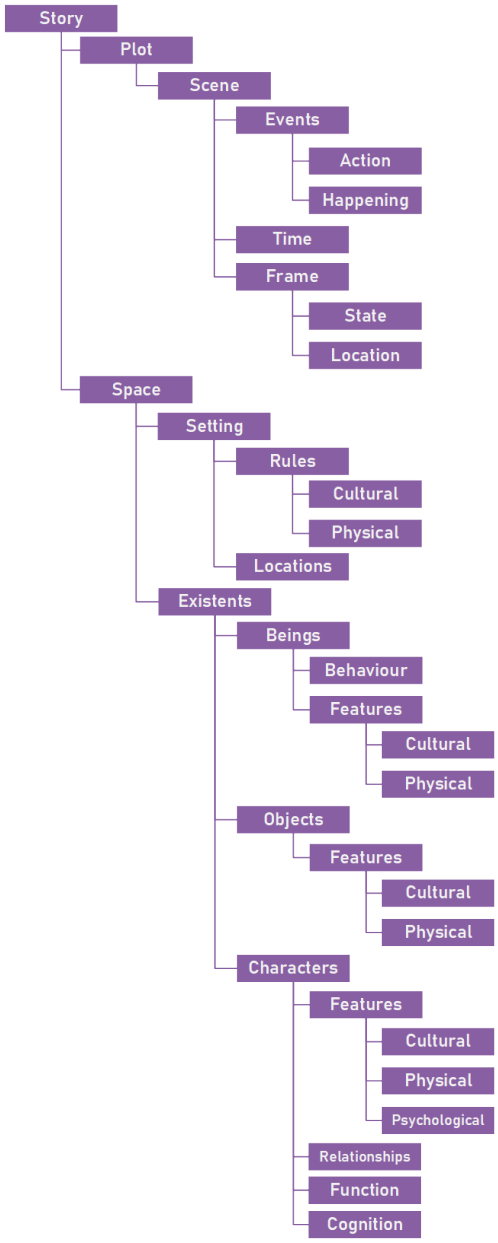


Figure 1. Stories common representation model.

The **space** encompasses the whole universe in which the plot is taking place and also all the places, beings and objects of which existence the characters are aware of, regardless of these elements are real or fictitious. The representation model considers that the space is composed by the setting and the existents. The **existents** are the whole set of actors that take a part in the story. They can be characters, living beings —an animal—, and an object in the setting. The two last types are mainly defined by their physical features and their cultural significance in the story. The **characters** are the most relevant, and also the most complex to represent, elements

in the story. The proposed model considers not only their physical, psychological and social features, but also their cognitive-related characteristics. The cognition of the characters is represented in a very detailed manner due to its importance for ensuring story consistency and characters liability. The aspects considered have been chosen after analysing those used by the existing storytelling systems [40, 30, 10, 29, 33, 37] and theoretical studies about Narrative [2, 31]. So, the representation of cognition includes the following facets:

- Goals: The goals are the results or achievements toward which the character effort is directed. The model considers two types of goals: conscious and unconscious. In the first case, the character is aware of them, in the second, they drive the character's actions, but he/she is not aware of them.
- Intentions: The intentions refer to the general plan that every character has, and the drive for his/her actions.
- Knowledge: Despite the characters act and interact in the same space, every single character could have different levels of knowledge concerning it. That means that the characters are not considered to be omniscient. This knowledge can evolve over the time, so characters can be acquiring or discarding knowledge as the story develops.
- Memories: Unlike the general knowledge, the memories refer to some past situations that have relevance in the story. For example, a memory can be referred to a past scene in which the character took part.
- Beliefs: The beliefs are a very subjective part of every character's cognition. They refer to facts about the world which the character considers as axioms, regardless of they are true. They can be part of the character's cultural or religious code, or simply originate in a particular misconception of the world.
- Dreams: The dreams represent the unconscious aspirations of the character. He/she may not be aware of them, but they can operate at a subconscious level and inspire his/her intentions.
- Fantasies: The fantasies are product of characters' imagination. They are beliefs or notions based on no solid foundation, a fact which the character is perfectly aware of. They represent aspirations that the character considers unreachable, but he/she enjoys thinking about them.
- Emotions: The emotions are related to the feelings of the character. They are usually influenced by the relationships that the character establishes with the others, and the evolution of them during the story.

Another relevant element of character's representation is the **function**. The idea is to provide a way of representing the main two approaches concerning the role of the characters in the plot. There are models that consider the plot as the result of characters interactions in a simulated story world, but there is another line of thought which considers that characters are subordinate to the narrative action. There are storytelling systems [20] that describe characters in terms of a structure based on their roles in the plot. Hence, the function tag refers to this approach and provides a way for linking the functional role of the character to the underlying structure of the story.

The **setting** is a combination of a set of physical —or virtual— locations in which the action of the story takes place, and the set of cultural and physical rules that govern the story world. The **locations** can be considered the scenario in which every scene that composes the plot takes place. So, as shown in the model, every scene links to its corresponding location.

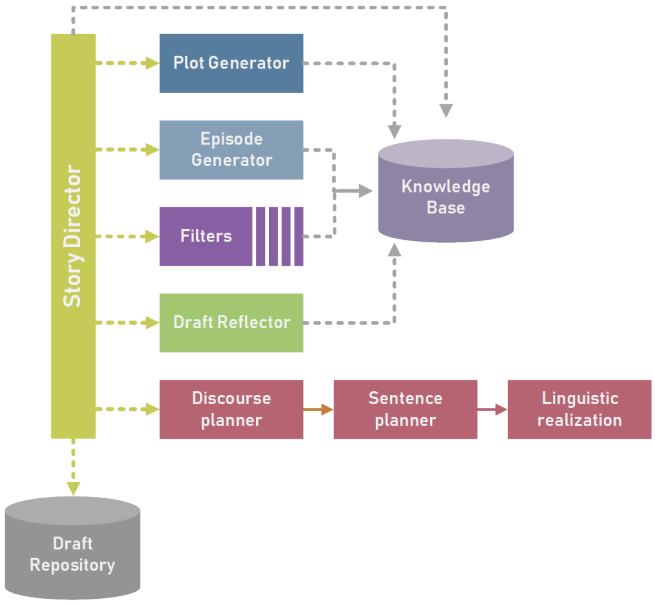


Figure 2. Architecture of Afanasyev.

4.2 Architecture of Afanasyev

The architecture of Afanasyev is based on a set of key microservices that provide the essential capabilities for story generation. Every microservice publishes an interface according to the REST model [13]. The joint operation of the microservices ecosystem is managed by the Story Director, which acts as an orchestrator of the services activity. It will request the APIs of the different services according to the steps of the generation process. This process will proceed iteratively, generating drafts that will be refined in each pass, until the established criteria for story completeness are met.

The main microservices in Afanasyev, depicted in Figure 2, are the following:

- Story Director
- Plot Generator
- Episode Generator
- Filter Manager
- Draft Reflector
- Discourse generation services (Discourse Planner, Sentence Planner and Linguistic Realization)

The key component of this framework is the **Story Director**, the inner architecture of which is depicted in Figure 3. It is strongly influenced by the Domain-Driven Design (DDD) principles [12].

The distinction between Application services and Domain services is precisely due to DDD. An application service has a clearly distinguishing role: it constitutes the environment for executing the domain logic, orchestrating the calls to the other components of the architecture: domain services, gateways and repositories. Domain services are only focused on performing domain logic which does not involve managing entities (Repositories) or calling external components (Gateways). So, they can rather be seen as components that provide procedural functionalities.

The Story Director has a clearly defined REST interface. The technical interface layer provides the logic necessary for implementing the communication-related requirements, allowing the isolation of

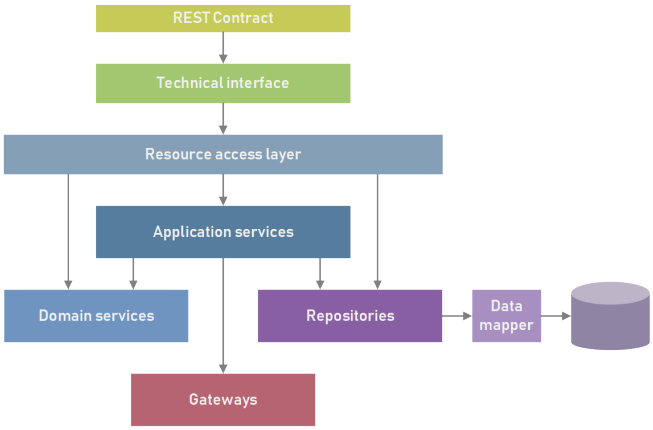


Figure 3. Story director architecture.

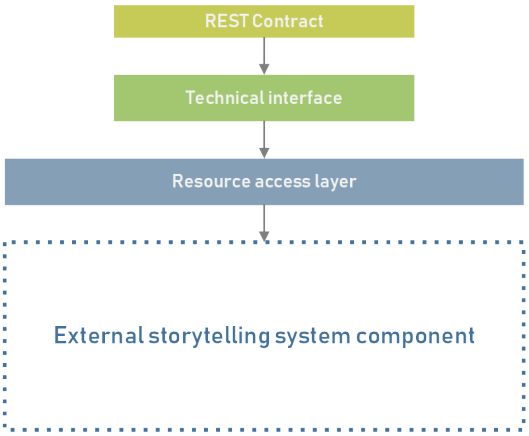


Figure 4. Marker microservices architecture.

the remaining components from them. The resource access layer provides a uniform interface for accessing the stories managed by the Story Director.

The repositories have been designed according to the Repository pattern [14], which provides a convenient abstraction for managing persisted objects. The inner database of the Story Director is an auxiliary store for persisting the life cycle of the ongoing drafts.

Persistence in Afanasyev is mainly composed by two stores: the Draft Repository and the Knowledge Base. The Draft Repository is a database that stores the ongoing drafts. The current implementation of this component is based on a NoSQL database[23] (MongoDB[1]). The knowledge base has the task of preserving all the knowledge related to concepts, relationships between concepts, rules, etc. It is a knowledge base generated from the contributions of the involved story generation systems. This model of knowledge syndication allows to increase the shared set of concepts each time a new system joins the ecosystem. Hence, every contributor performs an initial load expressing its rules by means of a controlled natural language expression. Namely, the current version counts on Attempto Controlled English (ACE) for this representation[17][16][27]. The use of a CNL for representing the knowledge allows the model to abstract from the programmatic representation used by each particular system, and to provide a greater robustness and consistency to the system architecture.

The **Plot Generator** main task is generating the complete plot structure. This includes the generation of the sequence of scenes that constitute the plot, the preconditions and postconditions that constrain every scene, and the articulation of the story in a high level.

The **Episode Generator** is in charge of developing the details of what happens in every scene of the plot. It must consider the preconditions and the postconditions defined for the scene by the Plot Generator, in order to create a scene detail that is consistent with them.

The **Filter Manager** is a service devoted to filter the population of generated drafts in order to select only the most promising stories, in terms of narrative tension or suspense. It is a very convenient tool for avoiding an explosion of irrelevant draft variants during the episode generation.

The **Draft Reflector** inspects the drafts for deciding if they are finished stories or if they must be improved in another iteration. For example, it checks if all the scenes of the plot have been detailed.

From a technical point of view, the **Plot Generator**, the **Episode Generator**, the **Filter Manager**, the **Draft Reflector** and the text generation services are basically marker microservices, with a predefined REST interface and a set of common architectural components. They are expected to be implemented by the particular story generation systems that collaborate in the generation process.

The internal architecture of these microservices, as Figure 4 shows, share partially the design of the Story Director. The components directly related to the intercommunication has been structured in the same way. They have a common layer for REST contract, with their corresponding technical interface, and the mandatory CNL mapping components. In their case, the resource access layer acts as an anticorruption layer[12] that isolates the inner logic of the service from the common framework infrastructure.

4.3 System operation

Afanasyev operates iteratively. Firstly, it generates a draft that will be completed by the various existing services in the architecture. The Story Director acts as the central component, orchestrating the requests to the different microservices. Table 1 summarizes the REST operations related to each microservice. The first step is always performed by the Plot Generator, which generates the basic structure of the plot. This provides a first basis for the story, with the sequence of scenes that make up the plot. Each scene is characterized by a previous state (precondition) and a later state (postcondition) of the world in which the action takes place. Every state is a collection of statements relating to the characters, living beings, and objects that exist in the story. In addition, each scene is associated with a specific setting. This setting is a reference to the list of existing settings defined in the story space.

Once the first draft is generated, the Story Director will persist it in the Draft Repository and then it will request the Episode Generator to generate the detail of what happens in each scene. For this, the Episode Generator receives as a parameter the draft, and the identifier of the scene that it must develop. Again, in this process the previous and final states of the scene are extremely important, since they will provide information to the Episode Generator about what can and can not happen in the scene. That is, the Episode Generator will only generate solutions for the scene that are coherent with the previous and final states, discarding the rest. The output will be a collection

Table 1. Afanasyev microservices operations summary.

Service	Method	Input	Output
Story Director	POST	Characters list, Pre/post spec	Story
Plot Generator	POST	Characters list, Pre/post spec	Draft
Episode Generator	PUT	Episode UUID, Draft	Draft
Filter Manager	POST	Episode UUID, Draft	Episode curves
Draft Reflector	POST	Draft	Draft Evaluation
Discourse planner	POST	Story	Text (NLG)

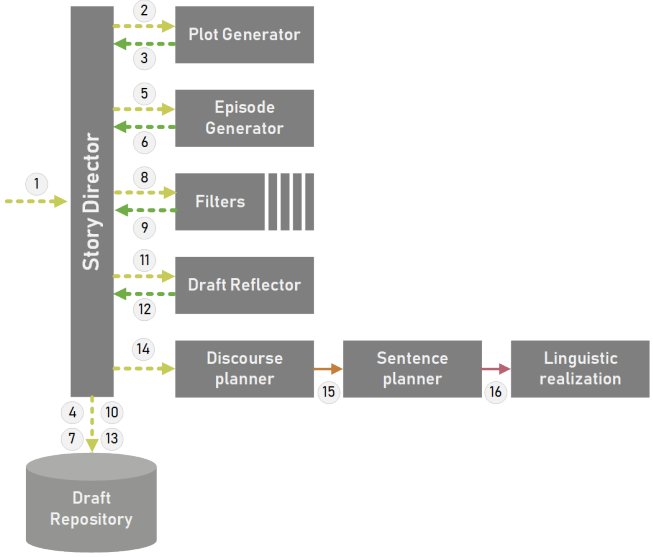


Figure 5. Operation of Afanasyev.

of possible continuations of the story, namely, a collection of drafts. Once again, every generated draft will be saved by the Story Director in the Draft Repository.

In the next step, the Story Director will request the Filter Manager to apply a sequence of filters on the generated drafts, and discard those considered as not promising. The number of filters is variable and they will always be applied in order, being the first the most important. Some of these filters can focus on aspects such as narrative tension or suspense. They allow us to make the stories more interesting by selecting those drafts that best fit the proposed parameters. The Story Director will remove the discarded drafts from the Draft Repository.

The final step in each iteration is provided by the Draft Reflector, which analyzes each of the drafts in progress and decides if the story has been completed, and therefore, stopping being a draft to become a finished story. The last step for the finished story is to generate the text in Natural Language. This task is performed by the discourse generation services, that work sequentially: Discourse Planner - Sentence Planner - Linguistic Realizer.

The whole operation of Afanasyev is summarized by Figure 5

The main advantage of this operation model is that the components of the architecture are basically slots that can be fitted by different services that follow different strategies. For example, the criteria for story completeness depend totally on the implementation of the Draft Reflector. Furthermore, the architecture admits the coexistence of various draft reflecting services that can be called by the Story Director according to higher order criteria. This feature provides a wider variety of behaviours during system operation.

5 DISCUSSION

Unlike previous approaches to collaborative story generation [35], Afanasyev is not geared towards the ad hoc integration of specific pre-existing systems, but rather to provide a general service-oriented framework that allows the construction of different storytelling systems by assembling components from various systems (or from only one, in the simplest case).

From an architectural point of view, Slant consists of a blackboard architecture [25] and a shared XML based story representation, which allows different storytelling systems or components to contribute to the story generation. This approach entails that every contributing system can access a shared working draft and enrich it. As part of the generation process, Slant provides mechanisms for selecting the most convenient contents in every iteration and deciding when to finish a ongoing story.

In contrast, in the service-based approach of Afanasyev, only the Story Director manages directly the ongoing drafts. The rest of the services can be invoked only according their interface and their operation is always orchestrated by the Story Director. This modularization, derived from the use of a microservices architecture, is not the only interesting feature. First of all, every service can be instantiated several times, and even exhibit different behaviour according to its configuration. For example, there can be several instances of the Plot Generator service, each with a different inner implementation, and the Story Director can request them to generate a draft in order to have a wider variety of plots. The same applies to the Episode Generator and the Draft Reflector services. In an API ecosystem, different versions of the same service can live together and be consumed independently. So, it would be possible to have an Episode Generator instance implemented from certain storytelling system, and another Episode Generator instance implemented from a different storytelling system.

Another interesting feature is that the architecture can be easily extended. The operation of every microservice in Afanasyev is completely independent from the others. If we wish to introduce a new microservice in the architecture, the only component that would require to be adapted would be the Story Director —in order to include this new service in the generation process that the Story Director manages.

Also, the Filter Manager service has been designed as an extensible sequence of filters that are applied in order to modify the draft received as a parameter. These filters are related to the degree of interest of the draft (for example, narrative tension and suspense). Adding a new filter simply requires to register the service that implements it into the Filter Manager.

Due to the coexistence of rules from various systems, it is assumed that there is no guarantee of consistency in the knowledge base. Achieving a full strict consistency would entail the validation of every new rule against the set of rules previously stored, and deciding which rule must be preserved in case of conflict. Another option would be the segmentation of the rules according to their origin as namespaces that would be locally consistent.

In the current version of Afanasyev it has been accepted that there can exist rules mutually inconsistent, even mutually exclusive (e.g.

"Magic does not exist" and "Magic exists"). The reason for this choice is to provide an open perspective during generation and leave it up to the human evaluator to decide whether the generated story is more interesting despite the potential inconsistencies.

A future option could be including non monotonic reasoning[15], providing default rules, or even developing truth maintenance mechanisms (e.g. "Magic does not exist for muggles"). These approaches are left for later as a future work due to their complexity and importance.

In addition to the above, the use of a domain-specific glossary would serve not only for establishing a proper definition of the knowledge domain, but also for reducing the risk of polysemy. One of the potential issues with CNL is that they are not specifically designed to address word sense disambiguation. The CNL are usually focused on analysing only the key words that are relevant for building the discourse representation structure, so it will be necessary to validate the portability of this representation over the different services.

6 CONCLUSIONS AND FUTURE WORK

The main advantage of the Afanasyev model comes from its modularity. By means of a flexible architectural structure, a common knowledge representation model and a set of services with well-defined interfaces, the proposed framework eases the development of collaborative story generation ecosystems. Some of these services might take the form of user interfaces to allow human intervention, so it also encourages the development of co-creation models.

In the present version of Afanasyev, for every draft processed in every iteration, there can be generated several continuations that are added to the population of drafts to process during the next iteration. On the generated population, a reflection process is applied by means of the Draft Reflector microservice, and the drafts that it considers already finished are marked as stories. This process continues until all drafts are marked as finished or a limit of iterations is reached (to guarantee completion). In the face of future work, the development of a service that helps to decide what is the most appropriate level of detail in each of the scenes is still pending. This aspect can be provided in a first instance by a human —applying a co-creation model—, but it would be perfectly evolved to introduce a component for automating this task.

In the short term, the next steps are focused on adding the capabilities of different existing storytelling systems such as Charade [33], STellA [30] and PropperWryter [20]. In a first approach, the goal is demonstrating the ability of the framework for reconstructing existing systems and the adequacy of the knowledge representation model for expressing the needs of various existing systems. Next, the objective would be the implementation of a real collaboration between different systems by mixing services from different origins.

ACKNOWLEDGEMENTS

This paper has been partially funded by the project IDiLyCo: Digital Inclusion, Language and Communication, Grant. No. TIN2015-66655-R (MINECO/FEDER).

References

[1] Mongodb official site. `https://www.mongodb.com/`, 2017. [Online; accessed 29-December-2017].
[2] Roland Barthes, *S/Z: an essay*, Siglo XXI, 1980.
[3] Seymour Benjamin Chatman, *Story and discourse: Narrative structure in fiction and film*, Cornell University Press, 1980.
[4] Eugenio Concepción, Pablo Gervás, and Gonzalo Méndez, 'An api-based approach to co-creation in automatic storytelling', in *6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017*, (2017).
[5] Eugenio Concepción, Pablo Gervás, and Gonzalo Méndez, 'A common model for representing stories in automatic storytelling', in *6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017*, (2017).
[6] Eugenio Concepción, Pablo Gervás, and Gonzalo Méndez, 'A microservice-based architecture for story generation', in *Microservices 2017*, (2017).
[7] Eugenio Concepción, Pablo Gervás, Gonzalo Méndez, and Carlos León, 'Using cnl for knowledge elicitation and exchange across story generation systems', in *International Workshop on Controlled Natural Language*, pp. 81–91. Springer, (2016).
[8] Eugenio Concepción, Gonzalo Mendez, and Pablo Gervás, 'Mining knowledge in storytelling systems for narrative generation', in *Proceedings of the INLG 2016 Workshop on Computational Creativity in Natural Language Generation*, pp. 41–50, (2016).
[9] Julio Cortázar, *Rayuela*, Editorial Sudamericana, Buenos Aires, 1963.
[10] Natlie Dehn, 'Story generation after tale-spin.', in *IJCAI*, volume 81, pp. 16–18, (1981).
[11] Thomas Erl, *Service-oriented architecture: a field guide to integrating XML and web services*, Prentice Hall PTR, 2004.
[12] Eric Evans, *Domain-driven design: tackling complexity in the heart of software*, Addison-Wesley Professional, 2004.
[13] Roy Thomas Fielding, *Architectural styles and the design of network-based software architectures*, Ph.D. dissertation, University of California, Irvine, 2000.
[14] Martin Fowler, *Patterns of enterprise application architecture*, Addison-Wesley Longman Publishing Co., Inc., 2002.
[15] Norbert E Fuchs, 'Reasoning in attempto controlled english: non-monotonicity', in *International Workshop on Controlled Natural Language*, pp. 13–24. Springer, (2016).
[16] Norbert E Fuchs, Kaarel Kaljurand, and Tobias Kuhn, 'Attempto controlled english for knowledge representation', in *Reasoning Web*, 104–124, Springer, (2008).
[17] Norbert E Fuchs, Kaarel Kaljurand, and Gerold Schneider, 'Attempto controlled english meets the challenges of knowledge representation, reasoning, interoperability and user interfaces.', in *FLAIRS Conference*, volume 12, pp. 664–669, (2006).
[18] Israel Gat and G Succi, 'A survey of the api economy', *Cut. Consort*, (2013).
[19] P. Gervás, 'Story generator algorithms', in *The Living Handbook of Narratology*, Hamburg University Press, (2012).
[20] Pablo Gervás, 'Propp's morphology of the folk tale as a grammar for generation', in *OASIcs-OpenAccess Series in Informatics*, volume 32. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, (2013).
[21] Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, and Raquel Hervás, 'Story plot generation based on cbr', *Knowledge-Based Systems*, **18**(4), 235–242, (2005).
[22] Joseph Goguen and D Fox Harrell, 'Style as a choice of blending principles', *Style and Meaning in Language, Art Music and Design*, 49–56, (2004).
[23] Jing Han, E Haihong, Guan Le, and Jian Du, 'Survey on nosql database', in *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, pp. 363–366. IEEE, (2011).
[24] D Fox Harrell, 'Walking blues changes undersea: Imaginative narrative in interactive poetry generation with the griot system', in *AAAI 2006 Workshop in Computational Aesthetics: Artificial Intelligence Approaches to Happiness and Beauty*, pp. 61–69, (2006).
[25] Barbara Hayes-Roth, *The blackboard architecture: A general framework for problem solving?*, Heuristic Programming Project, Computer Science Department, Stanford University, 1983.
[26] Sheldon Klein, 'Automatic novel writer: A status report', *Papers in text analysis and text description*, (1973).
[27] Tobias Kuhn, *Controlled English for knowledge representation*, Ph.D. dissertation, Faculty of Economics, Business Administration and Information Technology of the University of Zurich, 2009.
[28] Michael Lebowitz, 'Creating characters in a story-telling universe', *Poetics*, **13**(3), 171–194, (1984).
[29] Michael Lebowitz, 'Storytelling and generalization', in *Seventh Annual*

Conference of the Cognitive Science Society, pp. 100–109, (1985).
[30] Carlos León and Pablo Gervás, 'Creativity in story generation from the ground up: Nondeterministic simulation driven by narrative', in *5th International Conference on Computational Creativity, ICCC*, (2014).
[31] Uri Margolin, Peter Hühn, Jan Christoph Meister, John Pier, and Wolf Schmid, 'The living handbook of narratology', (2013).
[32] James R. Meehan, 'Tale-spin, an interactive program that writes stories', in *In Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pp. 91–98, (1977).
[33] Gonzalo Méndez, Pablo Gervás, and Carlos León, 'On the use of character affinities for story plot generation', in *Knowledge, Information and Creativity Support Systems*, 211–225, Springer, (2016).
[34] Nick Montfort, 'Curveship's automatic narrative style', in *Proceedings of the 6th International Conference on Foundations of Digital Games*, pp. 211–218. ACM, (2011).
[35] Nick Montfort, Rafael Pérez, D Fox Harrell, and Andrew Campana, 'Slant: A blackboard system to generate plot, figuration, and narrative discourse aspects of stories', in *Proceedings of the fourth international conference on computational creativity*, pp. 168–175, (2013).
[36] Mike P Papazoglou, 'Service-oriented computing: Concepts, characteristics and directions', in *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, pp. 3–12. IEEE, (2003).
[37] R. Perez y Perez, *MEXICA: A Computer Model of Creativity in Writing*, Ph.D. dissertation, The University of Sussex, 1999.
[38] Mark O Riedl and Robert Michael Young, 'Narrative planning: balancing plot and character', *Journal of Artificial Intelligence Research*, **39**(1), 217–268, (2010).
[39] Rolf Schwitter, 'Controlled natural languages for knowledge representation', in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pp. 1113–1121, Stroudsburg, PA, USA, (2010). Association for Computational Linguistics.
[40] Mei Si, Stacy C Marsella, and David V Pynadath, 'Thespian: Modeling socially normative behavior in a decision-theoretic framework', in *Intelligent Virtual Agents*, pp. 369–382. Springer, (2006).
[41] Scott R. Turner, *Minstrel: A Computer Model of Creativity and Storytelling*, Ph.D. dissertation, University of California at Los Angeles, Los Angeles, CA, USA, 1993. UMI Order no. GAX93-19933.
[42] Tony Veale, 'Creativity as a web service: A vision of human and computer creativity in the web era.', in *AAAI Spring Symposium: Creativity and (Early) Cognitive Development*, (2013).

4.5. Artículo: INES: A reconstruction of the Charade storytelling system using the Afanasyev Framework

Este artículo constituye el último eslabón de la cadena de trabajos de definición arquitectónica. INES es un sistema generador de historias que reescribe Charade (Méndez et al., 2016). El propósito de INES es probar la capacidad de Afanasyev como marco para la creación de sistemas generadores aplicándolo sobre un sistema preexistente y con una arquitectura original completamente distinta.

Gracias a que la arquitectura planteada en Afanasyev es flexible, INES no necesita implementar todos los microservicios definidos para ser operativo. En este caso, simplemente implementa un subconjunto del catálogo: el director de historias (*Story Director*), el generador de tramas (denominado *Audrey*), el generador de escenas (*Scene Generator*), un único filtro (en lugar del gestor de filtros completo), el revisor de borradores (*Draft Reflector*), y un sencillo generador de texto en lugar de todos los servicios de generación del discurso.

4.5.1. Cita completa

E. Concepción, Gervás, P., and Méndez, G. (2018) *INES: A reconstruction of the Charade storytelling system using the Afanasyev Framework*, in Ninth International Conference on Computational Creativity, ICCC 2018, Salamanca, Spain.

4.5.2. Resumen original

The present paper introduces INES (Interactive Narrative Emotional Storyteller), an instance of the Afanasyev story generation framework that rebuilds Charade, an agent-based storytelling system. The construction of INES pursues a double goal: to develop a more complete version of Charade, by including a plot generation stage; and to show the capability of Afanasyev as scaffolding for building united systems from sources of diverse kind. From a broad view, the resulting architecture is a microservice-oriented ecosystem in which every significant stage of the story generation process is implemented by a microservice that can be easily replaced by another, as long as the new microservice keeps the interface contract established by the Afanasyev model.

INES: A reconstruction of the Charade storytelling system using the Afanasyev Framework

Eugenio Concepción and **Pablo Gervás** and **Gonzalo Méndez**
Facultad de Informática
Instituto de Tecnología del Conocimiento
Universidad Complutense de Madrid
{econcepc,pgervas,gmendez}@ucm.es

Abstract

The present paper introduces INES (Interactive Narrative Emotional Storyteller), an instance of the Afanasyev story generation framework that rebuilds Charade, an agent-based storytelling system. The construction of INES pursues a double goal: to develop a more complete version of Charade, by including a plot generation stage; and to show the capability of Afanasyev as scaffolding for building united systems from sources of diverse kind. From a broad view, the resulting architecture is a microservice-oriented ecosystem in which every significant stage of the story generation process is implemented by a microservice that can be easily replaced by another, as long as the new microservice keeps the interface contract established by the Afanasyev model.

Introduction

Automatic story generation is a part of a wider research area in Artificial Intelligence named Computational Creativity (CC), which is the pursuit of creative behaviour in machines (Veale 2013).

A story generator algorithm (SGA) refers to a computational procedure resulting in an artefact that can be considered a story (Gervás 2012). The term story generation system can be considered as a synonym of storytelling systems, that is, a computational system designed to tell stories.

The operation of the story generation systems requires large amounts of knowledge. These systems are faced with a significant challenge of acquiring knowledge resources in the particular representation formats that they use. They meet an inherent difficulty when using formal languages in the detachment between the formulation of the needs in the real world and its representation in a formal construction. A possible solution can be the use of a Controlled Natural Language (CNL) for knowledge interchange (Concepción et al. 2016). This is precisely the approach introduced by the Afanasyev framework (Concepción, Gervás, and Méndez 2018). Afanasyev is a collaborative architectural model for automatic story generation which relates to a service-oriented architecture (Concepción, Gervás, and Méndez 2017a). It introduces an agnostic story representation model (Concepción, Gervás, and Méndez 2017b) that intends to ease the collaborative interchange of knowledge between different systems.

INES (Interactive Narrative Emotional Storyteller) is a reconstruction of Charade (Méndez, Gervás, and León 2016) based on the Afanasyev Framework. The original Charade system is a simulation-oriented agent-based story generation system. Charade was focused on generating stories about the evolution of the relationships between characters by running an unrestricted low-level simulation. The development of INES introduces a new stage in the Charade generation model, that is the plot generation. This stage provides the system with a more structured way of building the stories. Also, the development of INES allows for testing the suitability of the Afanasyev architectural structure and its knowledge representation model in a real-world context.

Background

The first story generation systems date back to the 1970s. The **Automatic Novel Writer** (Klein 1973) is considered to be the first storytelling system. It generated murder stories in a weekend party setting by means of generation grammars. **TALE-SPIN** (Meehan 1977) was another of the earlier story generators. It generated stories about the inhabitants of a forest. TALE-SPIN was a planning solver system that wrote up a story narrating the steps performed by the characters for achieving their goals. **Author** (Dehn 1981) was the first story generator to include the authors goals as a part of the story generation process. To this end, it intended to emulate the mind of a writer. From a technical point of view, Author also was a planner but, unlike TALE-SPIN, it used the planning to fulfill authorial goals instead of character goals. **Universe** (Lebowitz 1984) generated the scripts of a TV soap opera episodes in which a large cast of characters played out multiple, simultaneous, overlapping stories that never ended. In contrast with Author, Universe gave a special importance to the creation of characters, as it considered they were the driving force for generating stories. **Brutus** (Bringsjord and Ferrucci 1999) was a system that generated short stories using betrayal as leitmotiv. The main contribution of Brutus was its rich logical model for representing betrayal. This feature, along with its grammar-based generation component and its literary beautifier allowed it to generate quite complex stories. The **Virtual Storyteller** (Faas 2002; Swartjes 2006) is a Multi-Agent System that can generate stories by simulating a virtual world in which characters modeled by agents pursue their goals. In this way, the

story emerges from the events in the virtual world. **Fabulist** (Riedl and Young 2010) is a complete architecture for automatic story generation and presentation. Fabulist combines an author-centric approach together with a representation of characters intentionality.

Although there is not much specific literature on the subject, there are some noticeable efforts concerning the reconstruction of an existing story generation that have been carried out. **Minstrel** (Turner 1993) was a story generation system that told stories about King Arthur and his Knights of the Round Table. Each story was focused on a moral, which also provided the seed for developing the story. Minstrel was developed in Lisp (Berkeley and Bobrow 1966) and used an extension of a Lisp library called Rhapsody (Malkewitz and Iurgel 2006) for representing the knowledge required by the generation process.

Skald (Tearse et al. 2014) is a publicly-released rational reconstruction of Minstrel for analysing original Turner's work in search of new implications for future research. Skald is written in Scala, a functional programming language that runs over the Java Virtual Machine. It is based on a previous project named **Minstrel remixed** (Tearse et al. 2012), that tried to develop a collection of improvements over the original Minstrel. The original components of Minstrel, as described in Turner's dissertation (Turner 1993), are the starting point for the Skald design. The work developed in Skald can be considered not only a collection of enhancements over Minstrel but a globally different picture of the original Minstrel, as well as a new system that sets the stage for future research in story generation.

One of the Skald key findings is an improvement over Minstrel's limitations: the story library, story templates, and the recall system must be tailored to one another for the original system to function. Tearse (2012; 2014) shows that this can be mitigated through a number of techniques, such as adding differential costs to transformations to remove the least-successful author-level actions.

Although Skald contains a good number of enhancements over Minstrel remixed, these are all aimed to expand its capabilities in a few areas: transparency in story generation, exploration and measurement of the subtle workings of individual modules, improved stability, and better story output in terms of speed, size, and coherence.

Skald keeps the original Minstrel specifications, in the sense that it simulates the actions of a human when producing stories. Skald puts its novelty at a lower level, using different levels of modules for simulating different problems. For example, it uses low level simulations of problem solving processes, while authorial goals are simulated using modules in a higher level.

Materials and methods

Charade

Charade (Méndez, Gervás, and León 2014; 2016) models the relationship between two characters using their mutual affinities, and applies it for generating stories.

This system is an agent-based architecture developed using JADE (Java Agent Development Framework) (Bellifemine, Poggi, and Rimassa 1999). It consists of two types of agents: a Director Agent, that sets up the execution environment and creates the characters; and the Character Agents, one for each character of the story, whose interactions generate the story.

The main objective of the system was implementing an affinity model as decoupled as possible from the story domain, and testing it independently from other factors such as the environment in which the action takes place or the personality traits and emotional state of the characters. Due to this independence, it can be easily used to generate different kinds of stories.

The generator is based on a simulation of the characters' interaction. During the simulation, the characters perform actions that result in a variation of their affinity levels. According to the affinity level, the characters can be a couple, friends, indifferent, and enemies. Generation is independent of the domain; although, since it focuses on affinities, it works best in domains where this affinity makes sense. The simulation is not directed, so that it can not be considered to constitute a plot or a story by itself. The input includes a complete parametrization of possible actions, categorized according the type of relationship allowed for the characters, the simulated characters, and their relationships measured in terms of affinity. The output consists of a list of actions proposed by characters, and the response of their counterparts, that can accept or reject the proposals, with the variation of affinity between the characters involved. Despite no text being generated, it would be easy to use a template for generating a textual description.

Afanasyev

Afanasyev (Concepción, Gervás, and Méndez 2018; 2017a; 2017b) is a framework specifically designed for building service-based automatic story generation systems. From an architectural point of view, it is basically a collection of microservices orchestrated by a high-level service. Each service exposes their capabilities as REST-based APIs (Fielding 2000) and it understands and generates JSON messages. Due to the fact that the inner logic of any microservice can come from a different storytelling system, its interface must be adapted to match the required contract so the microservice can operate under the conditions specified by the framework. This is the reason why Afanasyev includes the definition of the common REST interfaces provided by the services and leaves to every particular system the details of the implementation.

The main microservices in Afanasyev, depicted in Figure 1, are the following:

- Story Director, the microservice that orchestrates the whole ecosystem.
- Plot Generator, the microservice that generates a high-level plot.
- Episode Generator, the microservice that fills the scenes that composed the plot.
- Filter Manager, the microservice that manages a set of filters that will be applied to the story each time it changes (due to the activity of the Episode Generator).

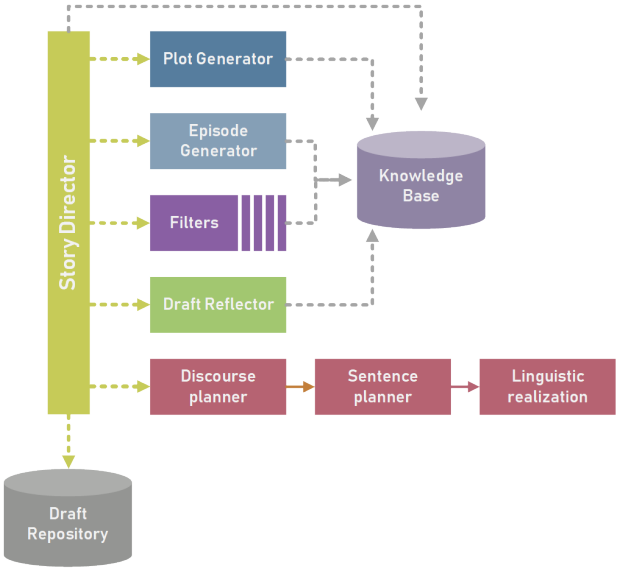


Figure 1: Architecture of Afanasyev.

- Draft Reflector, the microservice that analyses the story for deciding whether it is completed or not.
- Discourse generation services (Discourse Planner, Sentence Planner and Linguistic Realization), which turn the abstract story model into a human-readable text in Natural Language.

In order to allow the combined operation, the microservices of the framework require a common representation model for stories. The Afanasyev representation model (Concepción, Gervás, and Méndez 2017b) focuses on the knowledge that is directly related to the story, instead of that related to the generation process, which would be hard to export between different systems. The model has been designed as a hierarchical structure, in which the root concept is the **story**. Most of the leafs of this tree-like structure are assertions representing a piece of knowledge. These assertions are expressed by means of sentences in a Controlled Natural Language (CNL) (Schwitter 2010). In Afanasyev, every story is composed by a plot and a space. The plot represents the sequence of events —actions and happenings, that constitutes the skeleton of the story. The space encompasses the whole universe in which the story takes place, including the existents —characters, living beings and objects that take part in the story, and the setting —the set of locations mentioned in the story.

Persistence in Afanasyev is mainly composed by two stores: the Draft Repository and the Knowledge Base. The Draft Repository is a database that stores the ongoing drafts. The current implementation of this component is based on a NoSQL database (Han et al. 2011), namely MongoDB (2017). The knowledge base has the task of preserving all the knowledge related to concepts, relationships between concepts, rules, etc. It is a knowledge base generated from the contributions of the involved story generation systems. This model of knowledge syndication allows to increase the shared set of concepts each time a new system joins the ecosystem. Hence, every contributor performs an initial load expressing its rules.

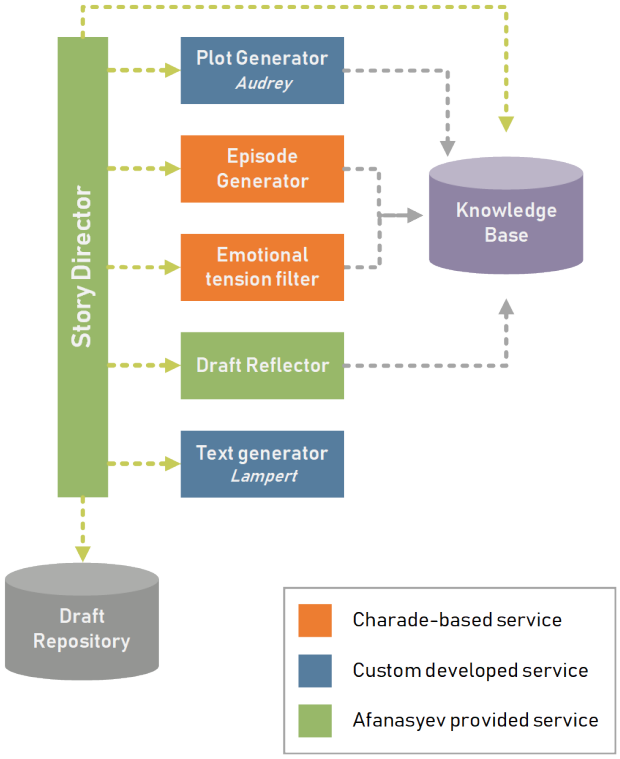


Figure 2: Architecture of INES.

INES

INES is the translation of the Charade storytelling system to the Afanasyev architectural framework. The purpose of this work is two-fold: to validate the capability of the Afanasyev model for supporting different story generation models and to prepare the integration of Charade in a wider service-based collaboration ecosystem.

The main adaptation work has focused on a central aspect of the original Charade behaviour: the directed simulation. In effect, Charade originally produced outputs that were the result of an unrestricted simulation. In the case of INES, there is a preexisting plot to which the output of every simulation must be adapted. This means that, for each scene, there is a specification based on precondition / postcondition that implies that not every possible result of the simulation is valid.

The architecture of INES, as adapted from Afanasyev, is depicted in Figure 2. It is a combination of Afanasyev ready-made services, along with the specific Charade adapted services and a set of newly created services, required by the framework:

- Story Director, provided by Afanasyev
- Plot Generator, required by Afanasyev and newly developed for INES

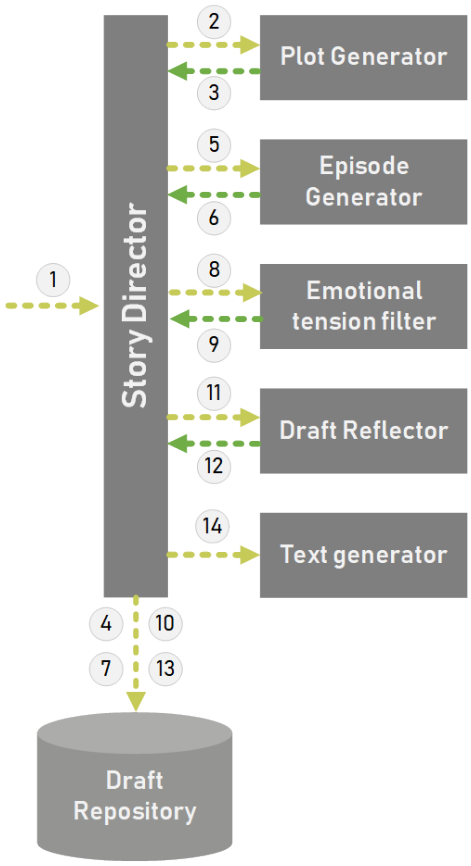


Figure 3: Operation of INES / Afanasyev.

- Episode Generator, created from the Charade system
- Emotional tension filter, created from the Charade system
- Draft Reflector, provided by Afanasyev
- Text generator, required by Afanasyev and newly created for INES

The Story Director

The architecture of Afanasyev is an ecosystem of microservices. The Story Director manages the joint operation of the whole ecosystem, as depicted in Figure 3. It orchestrates the execution of the different story generation stages by requesting the APIs of the different services. This processing proceeds iteratively, generating drafts that will be refined in each pass, until the established criteria for story completeness are met.

The first step consists in generating the basic structure of the plot. It is performed by the Plot Generator, that establishes the sequence of episodes that make up the plot. Each episode is interwoven with the others by means of its pre and post-conditions. These are collections of statements relating to the setting and the existents of the story.

The Plot Generator: Audrey

The **Plot Generator**, named **"Audrey"** —after Audrey Hepburn who played the lead role in "Charade", has been developed specifically for INES and is a template-based plot generator which produces outlines from a subset of the cinematographic basic plots compiled by Balló (Balló and Pérez 2007). Its basic procedure can be considered akin to those applied by systems like Gester (Pemberton 1989) and Teatrix (Machado, Paiva, and Brna 2001). The basic idea behind Audrey is building a story plot containing the main scenes that will be developed by the Charade-based Episode Generator. The plot building procedure starts by selecting one of the predefined templates, which consist of a conceptual structure with the shape of the plot. The template can be selected randomly or it can be picked according to the template name received as a parameter. Once a basic template is selected, Audrey gives it substance by instantiating the generic elements of such template. For achieving this, it requires to know about the context in which the story will be set. In this case, the context is inferred from the preconditions passed as parameters. These preconditions are a collection of assertions involving concepts that are necessarily kept in the knowledge base.

An example of one of these templates is "The destructive outsider". This story is essentially composed by the following episodes:

- The initial state: a peaceful community.
- The arrival of the outsider.
- The outsider acts against the members of the community, performing destructive actions, without being uncovered.
- The true evil nature of the outsider is revealed.
- The heroes rise from the community and fight against the outsider.
- The outsider is purged. The community becomes peaceful again.

In order to develop a consistent detail for every episode, Audrey requires a knowledge base that contains the main concepts presented in the plot. In this case, the plot mentions a "community", an "outsider", some "destructive actions" performed by the outsider, a group of "heroes" that rise against the outsider, and certain "purging actions" that the heroes perform. All these concepts are related to each other and can be represented by means of a graph. So, the required knowledge for instantiating the example is partially depicted in Figure 4.

So, the relationships between the concepts can be considered as assertions such as: "When the community is a family, then the outsider can be a new partner, an unknown relative and a new lover. When the outsider is a new partner, then the arrival can be a marriage".

The translation of these relationships to a physical database fits better with a graph-oriented database. In this particular case, the knowledge base has been implemented using Neo4j (Vukotic et al. 2014). So there are nodes with labels such as "Community", "Outsider", "Arrival" and "Action" representing the main plot concepts. The relationships

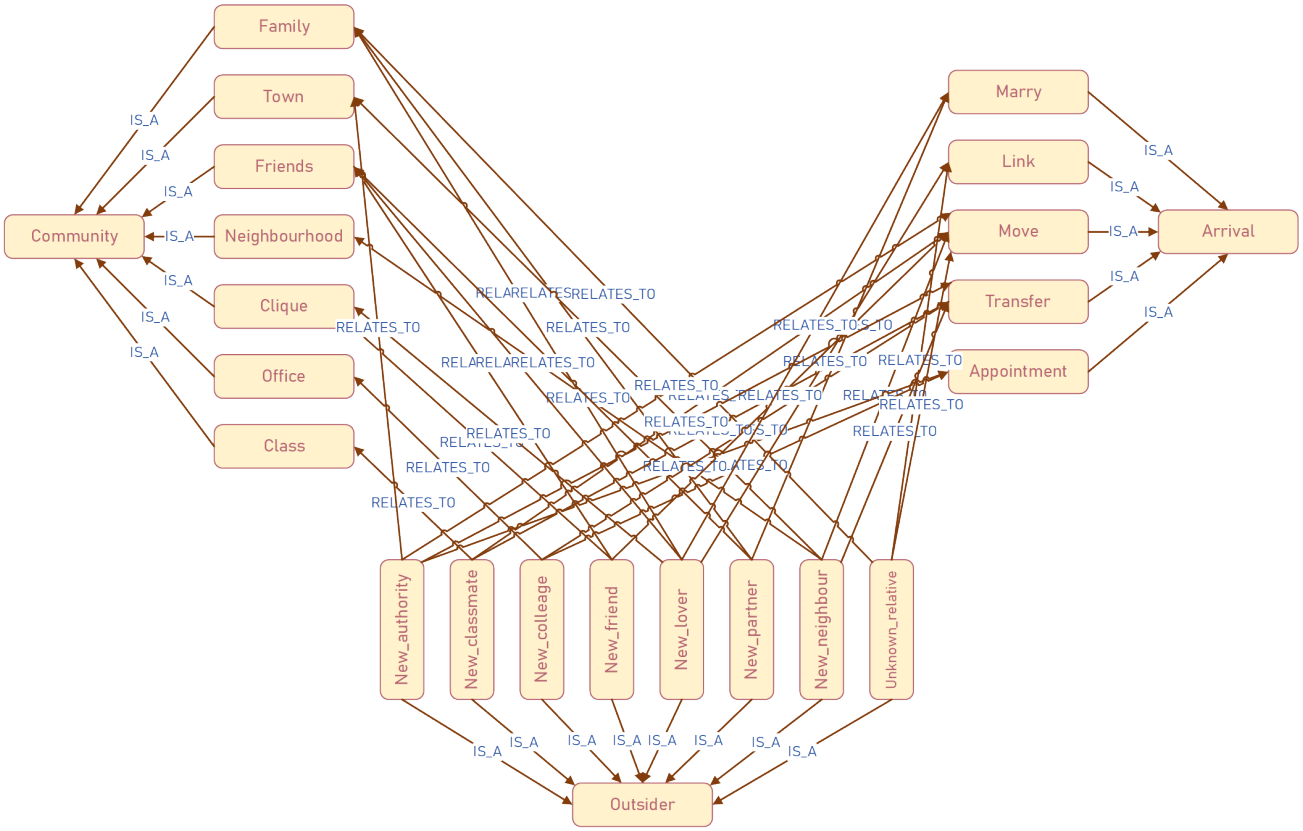


Figure 4: Partial view of the concepts relationships in the KB.

between the instances of the concepts are represented by means of the graph edges (i.e. database connections between nodes).

During the plot generation procedure, Audrey queries the knowledge base for extracting the possible instances for the concepts involved in the plot template. In this way, the concept "Outsider" is replaced by a "new neighbour" or a "new sheriff", according to the setting in which the story takes place.

The next step concerning this knowledge is to apply it for determining the actions that the characters can perform during the simulation in order to keep the story consistency. For example, the knowledge base can label the acts of "insult" and "kill" as "hostile actions". If the outsider has harmed the community by sowing discord, it would be unjustifiably excessive that the heroes reacted by killing him. In this case, the context of the story provides the episode generator with the appropriate actions that the actors could perform.

The Episode Generator

This microservice is based on the original Charade core. It generates a complete simulation of characters interaction according to the restrictions that are provided as input parameters. As mentioned in the previous section, the types of actions that can be consistently performed by the characters are limited by the context of the story. So, the episode generator receives not only the ongoing story, but also the preconditions and postconditions that the resulting simulation must match. This approach introduces a shift in the prior behaviour of the Charade's engine, which originally drove an unrestricted simulation.

Charade was designed for obtaining the list of possible actions from its configuration, during system startup. It distinguished between three types of actions (love, friendship and enmity). In order to ease the adaptation of Charade as a microservice in the Afanasyev ecosystem, the set of possible actions are passed to it as part of the request parameters. For selecting the most suitable actions, the Story Director queries the knowledge base and retrieves the context-related actions that better fit the storyline. For example, continuing the previous example, if the plot is related to a family and the outsider has stolen something, the actions carried out in response to this offence could be "insult", "report the burglary", "demand the restitution" and "demand to leave". These options would be retrieved and passed in the request as the proper actions that could be performed by the "heroes" of the story. Then, the episode generator would select some of them during the simulation and complete the detail of the episode. The current version of the episode generator preserves partially the randomness of the original Charade simulation model. In particular, it chooses randomly the actions performed by the characters from the

set of allowed actions.

Table 1 shows a sample story that can be generated by applying this model.

Episode	Actions
A peaceful community	John invites William to dinner John invites Mary to dinner William helps John to cook Mary gives a present to John
The arrival of the outsider	John makes a welcome party for David (the outsider) David gives a present to John William helps David to move Mary helps David to move
Outsider destructive actions	David steals a valuable object from John's house David tells Mary that William is the thief
Conflict	Mary believes David Mary insults William William gets angry with Mary
The outsider revealed	William discovers David stealing in John's house William tells John that David is the thief John tells Mary that David is the thief
The rise of the heroes	John insults David John demands David to leave David leaves the town
Conclusion	Mary says sorry to William William gives thanks to John Mary gives thanks to John

Table 1: A sample story based on "The destructive outsider"

The Emotional tension filter

The current version of the Emotional tension filter works in a very simple way. It is a filter which is invoked after every episode simulation —performed by the Episode Generator, and it determines if the generated actions fit certain drama parameters. To meet this purpose, the Emotional tension filter considers the semantic information associated to the actions in the knowledge base to adjust the strength of the drama in the story. For example, considering again the story of "The destructive outsider" plot, an action such as "to slap" the outsider is much more dramatic than "to demand him to leave". By establishing the threshold for the tension, this service helps the Story Director to select the most dramatic continuation of the plot. So, this filter removes a subset of the generated episodes, and makes the Story Director to call again the Episode Generator until the whole plot has been adequately completed.

All the actions referenced in the knowledge base have a numerical attribute which reflects its intensity in terms of drama. This is a feature closely related to the original Charade operation (Méndez, Gervás, and León 2016; 2014). The higher the intensity of the action is, the higher is the numerical value. This representation helps the filter to decide whether an episode deserves to be included in the draft or not.

The Draft Reflector

The Draft Reflector of INES is the original basic Afanasyev-provided Draft Reflector. This microservice simply checks if all the episodes have been developed according to the plot restrictions. In this case, the choice is based on the need of keeping the draft analysis stage as simple as possible. The interest of the INES model is related to the ability of Afanasyev to provide a suitable architecture for building a system like Charade by means of its building blocks.

The Text Generator: Lampert

The other INES-specific service developed is the Text Generator, named "Lampert" —after Audrey Hepburn character's surname in "Charade". Lampert is microservice that translates the plot, represented as a data structure, into a text in Natural Language. Its core is based in the SimpleNLG Java library (Gatt and Reiter 2009).

The text generation is the last stage in the story generation process. Lampert has been designed simply for conveying the story represented in the Afanasyev common representation. Its purpose is not so much being a literary beautifier but providing a human-readable summary of the story.

Discussion

Afanasyev is not focused towards the ad hoc integration of specific pre-existing systems, but rather to provide a general service-oriented framework that allows the construction of different storytelling systems by assembling components from various systems (or from only one, in the simplest case). For this reason, the adaptation of Charade has required a number of transformations. Firstly, Charade has been designed as a lightweight agent-based architecture. Its essential logic has been preserved in INES, but the system operation has been restructured. The operation of every microservice in INES is completely independent from the others. Adapting the simulation flow has involved the development of a couple of new components that did not exist in Charade: the Plot Generator and the Text Generator. These two microservices could be easily replaced by other microservices based on different approaches that the current ones, as this is the essence of the Afanasyev framework.

The generation model of the Plot Generator is quite simple, but also very convenient for filling the existing gaps in the original model. It can be easily extended by providing more plot structure templates. Also, the richer the knowledge base is, the more interesting the generated stories are. As it has been shown, the role of the knowledge base is essential in this model for achieving coherent and believable stories. The same basic plot template can be instantiated in a wide spectrum of stories. As new instances are added to the database, the variability will increase accordingly.

Another relevant addition to the original Charade behaviour is the Emotional Tension Filter. It allows the system to generate stories with a greater drama, or not, depending on the filtering values. This service can be enhanced,

or even replaced by a much more complex one, in order to help to create stories according to certain narrative tension curves. This configuration will entail a more global way of operation, considering not only the particular tension of an episode, but the evolution of the whole narrative arc.

Despite its simplicity, the Text Generator service provides a useful output. It has been deliberately designed for providing a summary in Natural Language rather than an elaborate literary text. Naturally, it can also be replaced by a much more complex surface realizer which provides a more polished literary work. A future candidate could be the TAP SurReal Surface Realizer (Hervás and Gervás 2009).

Conclusions and future work

The Afanasyev framework, despite having been originally conceived as an architectural model for building collaborative storytelling architectures, should not be seen solely as a tool for system integration. The purpose of developing INES was to prove that the Afanasyev framework can also be used for rebuilding any system as a microservice-based model. In this particular case, the adaptation of a purely agent-based simulation-oriented story generation system to a microservice-based pre-existing framework was particularly challenging. The resulting system can be considered an evolved version of the original Charade system, with a more structured approach to story generation.

Another interesting derivative of the work carried out during the design and development of INES is the knowledge base itself. It has been addressed using a representation model based on a graph-oriented database. This has allowed for simplifying the representation, as well as to use a general industry-oriented development stack, instead of a stack specifically oriented to Artificial Intelligence. The fact that the database can also be consulted by means of a REST interface provides an additional decoupling mechanism that will allow to evolve it independently, and even its replacement, without affecting the operation of the rest of the microservices ecosystem.

In the present version of INES, for every draft processed in every iteration, several continuations can be generated and added to the population of drafts to process during the next iteration. On the generated population, a reflection process is applied by means of the Draft Reflector microservice, and the drafts that it considers already finished are marked as stories. This process continues until all drafts are marked as finished or a limit of iterations is reached (to guarantee completion). In the face of future work, the development of a service that helps to decide what is the most appropriate level of detail in each of the scenes is still pending. This aspect can be provided in a first instance by a human —applying a co-creation model—, but it would be perfectly evolved to introduce a component for automating this task.

Acknowledgments

This paper has been partially funded by the projects IDiLyCo: Digital Inclusion, Language and Communication, Grant. No. TIN2015-66655-R (MINECO/FEDER) and InVITAR-IA: Infraestructuras para la Visibilización, Integración y Transferencia de Aplicaciones y Resultados de Inteligencia Artificial, UCM Grant. No. FEI-EU-17-23.

References

Balló, J., and Pérez, X. 2007. *La semilla inmortal: los argumentos universales en el cine*. Ed. Anagrama.

Bellifemine, F.; Poggi, A.; and Rimassa, G. 1999. Jade–a fipa-compliant agent framework. In *Proceedings of PAAM*, volume 99, 33. London.

Berkeley, E. C., and Bobrow, D. G. 1966. *The programming language LISP: Its operation and applications*. MIT Press.

Bringsjord, S., and Ferrucci, D. 1999. *Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine*. Psychology Press.

Concepción, E.; Gervás, P.; Méndez, G.; and León, C. 2016. Using cnl for knowledge elicitation and exchange across story generation systems. In *International Workshop on Controlled Natural Language*, 81–91. Springer.

Concepción, E.; Gervás, P.; and Méndez, G. 2017a. An api-based approach to co-creation in automatic storytelling. In *6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017*.

Concepción, E.; Gervás, P.; and Méndez, G. 2017b. A common model for representing stories in automatic storytelling. In *6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017*.

Concepción, E.; Gervás, P.; and Méndez, G. 2018. Afanasyev: A collaborative architectural model for automatic story generation. In *5th AISB Symposium on Computational Creativity. AISB 2018*.

Dehn, N. 1981. Story generation after tale-spin. In *IJCAI*, volume 81, 16–18.

Faas, S. 2002. Virtual storyteller: an approach to computational storytelling. *Unpublished masters thesis, University of Twente, Department of Electrical Engineering, Mathematics and Computer Science*.

Fielding, R. T. 2000. *Architectural styles and the design of network-based software architectures*. Ph.D. Dissertation, University of California, Irvine.

Gatt, A., and Reiter, E. 2009. Simplenlg: A realisation engine for practical applications. In *Proceedings of the 12th European Workshop on Natural Language Generation*, 90–93. Association for Computational Linguistics.

Gervás, P. 2012. Story generator algorithms. In *The Living Handbook of Narratology*. Hamburg University Press.

Han, J.; Haihong, E.; Le, G.; and Du, J. 2011. Survey on nosql database. In *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, 363–366. IEEE.

Hervás, R., and Gervás, P. 2009. Evolutionary and case-based approaches to reg: Nil-ucm-evotap, nil-ucm-valuescbr and nil-ucm-evocbr. In *Proceedings of the 12th European Workshop on Natural Language Generation*, 187–188. Association for Computational Linguistics.

Klein, S. 1973. Automatic novel writer: A status report. *Papers in text analysis and text description*.

Lebowitz, M. 1984. Creating characters in a story-telling universe. *Poetics* 13(3):171–194.

Machado, I.; Paiva, A.; and Brna, P. 2001. Real characters in virtual stories. In *International Conference on Virtual Storytelling*, 127–134. Springer.

Malkewitz, S. G. R., and Iurgel, I. 2006. Technologies for interactive digital storytelling and entertainment. In *TIDSE*. Springer.

Meehan, J. R. 1977. Tale-spin, an interactive program that writes stories. In *In Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 91–98.

Méndez, G.; Gervás, P.; and León, C. 2014. A model of character affinity for agent-based story generation. In *9th International Conference on Knowledge, Information and Creativity Support Systems, Limassol, Cyprus*, volume 11, 2014.

Méndez, G.; Gervás, P.; and León, C. 2016. On the use of character affinities for story plot generation. In *Knowledge, Information and Creativity Support Systems*. Springer. 211–225.

2017. Mongodb official site. `https://www.mongodb.com/`. [Online; accessed 29-December-2017].

Pemberton, L. 1989. A modular approach to story generation. In *Proceedings of the fourth conference on European chapter of the Association for Computational Linguistics*, 217–224. Association for Computational Linguistics.

Riedl, M. O., and Young, R. M. 2010. Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research* 39(1):217–268.

Schwitter, R. 2010. Controlled natural languages for knowledge representation. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, 1113–1121. Stroudsburg, PA, USA: Association for Computational Linguistics.

Swartjes, I. 2006. The plot thickens: bringing structure and meaning into automated story generation.

Tearse, B. R.; Mawhorter, P. A.; Mateas, M.; and Wardrip-Fruin, N. 2012. Lessons learned from a rational reconstruction of minstrel. In *AAAI*.

Tearse, B.; Mawhorter, P.; Mateas, M.; and Wardrip-Fruin, N. 2014. Skald: minstrel reconstructed. *IEEE Transactions on Computational Intelligence and AI in Games* 6(2):156–165.

Turner, S. R. 1993. *Minstrel: A Computer Model of Creativity and Storytelling*. Ph.D. Dissertation, University of California at Los Angeles, Los Angeles, CA, USA. UMI Order no. GAX93-19933.

Veale, T. 2013. Creativity as a web service: A vision of human and computer creativity in the web era. In *AAAI Spring Symposium: Creativity and (Early) Cognitive Development*.

Vukotic, A.; Watt, N.; Abedrabbo, T.; Fox, D.; and Partner, J. 2014. *Neo4j in action*. Manning Publications Co.

4.6. Artículo: The long path to narrative generations

Este artículo es un compendio de la evolución de los esfuerzos por desarrollar técnicas de generación. Recoge una versión resumida de los mecanismos empleados por STellA (León y Gervás, 2014), Charade (Méndez et al., 2016), PropperWriter (Gervás, 2016) y Afanasyev (Concepción et al., 2018a).

4.6.1. Cita completa

P. Gervás, E. Concepción, C. León, G. Méndez and P. Delatorre (2019) *The long path to narrative generation*, in IBM Journal of Research and Development, vol. 63, no. 1, pp. 8:1-8:10.

4.6.2. Resumen original

Narrative generation, understood as the task of constructing computational models of the way in which humans build stories, has been shown to involve a number of separate processes, related to different purposes to which it can be applied, and focusing on specific features that make stories valuable. This paper reviews a set of story generation systems developed by the authors of this contribution, each focusing on different aspects and functions of stories. These systems provide an initial breakdown of how the term "storytelling" might be either instantiated or broken down into component processes. The systems cover functionalities such as generating valid plot structures, simulating character's behaviors or the evolution of affinities between them, either reporting or fictionalizing events observed in real life, and revising a story draft to maximize the suspense it induces in its readers. These functionalities are not intended to exhaust the set of possible operations involved in storytelling, but they constitute an initial set to understand the complexity of the task. The paper also includes two proposals—one theoretical and one technological—for understanding how a set of such functionalities might be composed into a broader operational process that produces more elaborate stories.

The long path to narrative generation

P. Gervás
E. Concepción
C. León
G. Méndez
P. Delatorre

Narrative generation, understood as the task of constructing computational models of the way in which humans build stories, has been shown to involve a number of separate processes, related to different purposes to which it can be applied, and focusing on specific features that make stories valuable. This paper reviews a set of story generation systems developed by the authors of this contribution, each focusing on different aspects and functions of stories. These systems provide an initial breakdown of how the term "storytelling" might be either instantiated or broken down into component processes. The systems cover functionalities such as generating valid plot structures, simulating character's behaviors or the evolution of affinities between them, either reporting or fictionalizing events observed in real life, and revising a story draft to maximize the suspense it induces in its readers. These functionalities are not intended to exhaust the set of possible operations involved in storytelling, but they constitute an initial set to understand the complexity of the task. The paper also includes two proposals—one theoretical and one technological—for understanding how a set of such functionalities might be composed into a broader operational process that produces more elaborate stories.

Introduction

Storytelling is currently perceived as a fundamental component of the human cognitive ability [1] and as a crucial tool for successful participation in modern society [2]. It was also one of the earliest goals that artificial intelligence set for itself when it first started trying to model human abilities [3] and remains an active field of research to this day [4]. In terms of the pragmatic definition of computational creativity as the "*engineering of computational systems which, by taking on particular responsibilities, exhibit behaviors that unbiased observers would deem to be creative*" [5], storytelling has significant potential to contribute to the field.

From a pragmatic point of view, the task of modeling the human storytelling process constitutes an engineering challenge of the first order. Whereas successful prototypes of the storytelling ability have been achieved by employing individual technologies—such as planning [6], case-based reasoning [7], or agent-based modeling [8] to cite a few examples—it is unlikely that such a complex human ability can ever be modeled fully by recourse to a single technology. The fact that particular technologies lead to valuable results—by capturing some of the features that make up a valid story—clearly suggests that those technologies have something to contribute to the task. Yet the solutions in each case also show shortcomings with respect to features not specifically addressed by the technology in question.

The hypothesis underlying the material presented in this paper is that each of these approaches focuses the modeling of storytelling on a particular subtask that is quite capable of producing stories with valuable features. Some of the features considered include the different views of stories, such as narrative structures, simulations, evolving networks of characters affinity, narrations of observed facts, or suspense-driven entertainment. An important corollary of this hypothesis is that human storytellers rely on a combination of the subtasks based on these different features to obtain rich stories that combine the various features in rounded whole.

This paper outlines a number of significant insights into the processes required to inject "narrative" quality into machine-enabled communication. This section introduces the problem and describes the purpose of the paper. The second section outlines the different approaches that have

Digital Object Identifier: 10.1147/JRD.2019.2896157

been followed in storytelling research. The third section presents a number of alternative approaches for the consideration of stories, each one of them addressed in specific research systems to model different aspects of narrative communication. The fourth section addresses the challenge of integrating these subtasks into a single operational framework. This is done from two different points of view, one theoretical—a conceptual analysis of narrative communication into a set of processes that interact in different ways to give rise to the different modes of communication described above—and one computational—the Afanasiev software engineering framework for allowing the composition of the computational systems described above.

Existing approaches to narrative generation

There has been a significant number of efforts in the past to develop systems capable of generating stories, and an exhaustive review of them is beyond the scope of this paper. Interested readers are invited to consult some of the existing surveys [3, 4]. To provide a context for the set of different approaches considered here, this section reviews how the existing efforts in story generation have been categorized by prior authors.

An initial classification was provided by Bailey [9], who considered four different approaches to storytelling, depending on the perspective under which stories were addressed: author models—focusing on processes applied by authors; story models—focusing on structural properties of stories; world models—focusing on world simulation; and reader models—focusing on the effect of stories on the reader. Gervás et al. [10] proposed a classification of story generators based on the particular artificial intelligence technologies they employ, and identified generators based on planning and generators based on grammars. O'Neil [11] distinguished between generators based on searching over the set of possible sequences of actions and generators based on adapting existing stories into new versions to match a new purpose. Niehaus [12] established a distinction between systems that simulate a world from which the story is chosen and systems that operate by deciding among a set of narrative elements such as plot structure, character dynamics, or the experience of the reader.

Kybartas and Bidarra [4] review systems from a wider perspective, allowing for collaboration between a human author and a computer system. In this context, they classify systems in terms of the degree to which the features under consideration are automated or left to be done manually by the human user. Within this view, they address as features *plot*, the structure of the story, and *place*, the world that underlies the story.

In general terms, the variety of contrasting analysis suggests that there are many relevant aspects to story generation, and that different attempts at classification focus on specific ones while relegating others to secondary roles. However, successful modeling of human storytelling abilities is likely to require explicit treatment of all of these aspects in an integrated manner [13].

Deconstructing storytelling

One of the inspiring goals of research on storytelling has been to identify and develop mechanisms of interaction between people and machines that exploit language as means for human communication. Some of the solutions considered for this purpose have been based on Natural Language Processing, using various technologies for knowledge representation, and supported by a myriad of techniques from the domain of Artificial Intelligence. These mechanisms have been applied in practical contexts including intelligent information access, automated processing of medical documents, domotic interfaces, and verbal guidance for visually impaired users.

Although the empirical validation of these solutions was consistently positive, the solutions were often perceived as not reaching the level of fluency and naturalness that would be expected from a human carrying out the same tasks. However, it was difficult to identify what features were missing in the automated solutions that would have made them closer to human performance.

The progressive increase of public awareness of the importance of storytelling as a fundamental communication tool [1, 2] suggested that the solutions that had been attempted were lacking a "narrative" quality that humans tend to impart to their communication efforts when they are directed to other humans. As a consequence, the research focus in the world of computational storytelling is progressively shifting to consider the role of narrative in communication and to explore how this role might be modeled in computational terms.

However, the "narrative" quality that endears human-originated linguistic communication to human recipients is at the same time much less and much more than generating a story. Less because, in many cases, there is no need to invent a story to tell, but rather to "dress up" a given set of facts as a story, to use a known story as scaffolding for a content to be communicated, or to phrase a set of dry facts as if they were part of a story. More because, in other cases, it will involve inferring the best possible story implicit in a set of facts, or reading a story and finding a way to rephrase it that makes it more entertaining.

In this way, the challenge faced by a researcher hoping to build computational models of this kind of communication goes beyond the generation of stories and requires understanding of how the stories are received, interpreted, and validated. If stories are to be used as a code in a communication context, algorithms for encoding facts into stories need to be backed up with some notion of how the stories are decoded back into facts. For all of these

processes, it becomes important to know what features are necessary for a discourse to be considered a story and what makes a particular story "better" than another.

Addressing specific subtasks of storytelling

The work of modeling the human storytelling task has been addressed by the authors less in terms of which existing technology might fare well as a single actor in the task and more in terms of which particular feature of stories should be given priority in a given implementation. As a result of this policy, several different systems have been developed.

Stories as narrative structures

One of the most visible components of stories is its narrative structure or plot. It is also a component on which there is a large corpus of theoretical work, both from the field of narratology and from the field of creative writing. Our most successful attempt to model narrative structure as a driving force for a story generator is the PropperWryter system.

PropperWryter is a program that generates the narrative structure for a single plot line, described in terms of a vocabulary of abstract representations of events that may happen in such a plot. It evolved from the Propper system [14], which generated plot structures for Russian folk tales based on Propp's Morphology of the Folktale [15]. Propp identified a set of regularities across a corpus of Russian folk tales in terms of character functions, understood as acts of the character, defined from the point of view of their significance for the course of the action.

The PropperWryter system is composed of a set of modules: A plot driver generator builds the sequence of Proppian character functions that determine the structure of the tale, a fabula generator instantiates the character functions in the resulting sequence with particular story actions, a casting module assigns particular characters to the arguments of the selected story actions, and a textual rendering module converts the final conceptual plan for a story into text.

An extension of the PropperWryter software as a generator of plot lines for musical theater pieces was used for the composition of *Beyond the Fence*, the first-ever experimental computer-generated musical. *Beyond the Fence* opened successfully at the Arts Theatre in London on February 22, 2016, and enjoyed a successful two-week run [16].

Stories as simulations

A different point of view on stories is to consider them as tools for communicating the evolution of a given world. To model this view, the Story TELLing Algorithm (STellA system) [17] builds stories from a set of simulations of what happens in the world.

STellA was designed as a take on the commonsense knowledge bottleneck that constrains the possibilities of the existing story generation systems. Instead of focusing on one specific aspect of narratology or some concrete technology at hand to be tested for story generation, the design efforts in STellA were put on constructing a strong framework and an implementation for providing a knowledge-intensive solution.

In STellA, plots are assumed to be the result of very developed and complex interactions between characters and the world. As such, a detailed simulation takes place behind the scenes in order to produce a rich, complex material onto which to draw the narrative plot.

Therefore, two generative layers were carried out in STellA: First, the simulation layer applies physical, social, and mental rules in a nondeterministic way. This means that many possible simulations are produced. The vast space of plots is a basic source of creativity, but this expansion is unconstrained, and many nonvaluable stories are produced. The amount of "raw creativity" was approximated with a specific metric, the *entropy*, which reflects the amount of invented information in the simulation.

This exhaustive generation required several models running sequentially (and nondeterministically, for those models allowing for it): a physical ground for objects, collisions, and movement; an agent layer for planning; a layer for describing emotions, grabbing things, attacking, loving, etc. While most of these layers were implemented according to not very complex models, connecting them into a logic process required a relatively high effort.

All this detailed simulation was enough to provide many coherent action sequences between characters and the environment surrounding them, but these sequences of events do not make an interesting narrative by themselves. STellA needed a new level of control over the generation in order to constrain the exploration and let the user set a number of input parameters for the system to produce a particular kind of narratives.

The identification of valuable plots was carried out by the *narrative layer*, which received partial simulations, discarded the nonpromising ones, and set parameters for subsequent expansion. The narrative layer used explicit constraints, user objectives, and *narrative curves* [18] to drive story generation. These curves measure and restrict the amount of suspense, action, or entropy a simulation has introduced. **Figure 1** shows an example of curve evolution in STellA.

The design of the knowledge base not only affected the exploration of states, but also the development of the narrative layer itself. Providing models for specific curves, especially in terms of the knowledge that STellA used to describe the plots, is a hard engineering problem in its own right. Identifying the amount of suspense in a certain point of the plot, the perception of danger or the evolving

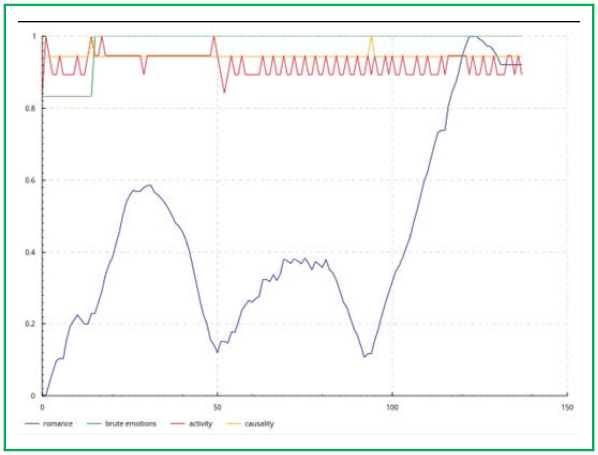


Figure 1

Curve evolution in STellA. As the story advances, the curves represent its narrative features.

romanticism between a couple required both the identification of plausible models and the implementation in terms of the underlying representation.

STellA was successful, and the implementation was capable of generating a massive amount of semantically valid stories. The costly engineering process of putting together many subsystems, plus the design challenge of a big architecture for nondeterministic generation, is high, but STellA was able to produce a huge set of plots.

On the other hand, the exhaustive exploration was an intractable problem. The amount of plots that the system could generate was big, but the real problem is that there are no models for identifying when an *optimal* story was produced. Validity, coherence, user objectives, and a shallow approach to narrative quality were implemented, but a real metric for *value* or *novelty* was not achieved.

Stories as evolving networks of character affinity

Stories may also be considered as reports on the evolution of a set of characters and the relations between them. The Charade system addresses this way of understanding stories.

Charade is a storytelling system that relies on a character simulation in order to generate the stories [19]. Charade builds upon Goethe's theory of *elective affinities* [20] to represent human relations, which shows how affinities between the characters of a story can be represented by a topological chart. This theory was subsequently explored by Polti [21], who out the assertion made by Gozzi (the author of *Turandot*) saying that there could only be 36 dramatic situations. After analyzing these situations, their variations, and the characters involved, Polti concludes that the subtle differences between these situations have to do with "*the ties of friendship or kinship between the characters.*" A similar approach is used in *Thespian* [22], the social behavior framework used to develop the *TLCTS* system [23].

The main objective of the Charade system is to study how character affinities can be used in the storytelling systems in order to endow characters with believable interactions that can be used to enrich certain episodes within the plot. In order to achieve this goal, we have modeled four different levels of affinity: foe (no or low affinity), indifferent (slight affinity), friend (medium affinity), and mate (high affinity). We have used a fuzzy approach in order to model affinities, similarly to what is done in other cognitive or emotional architectures [24, 25]. The advantage over a crisp model is that it allows us to overlap the affinity levels on their limits, which prevents relationships from changing constantly when moving around the limits of two different levels.

An additional feature of this affinity model is that it is not symmetrical. For any two characters, their mutual affinity is likely to have different values, and it may even be situated in different levels, with the exception of mates. However, if they are not mates, one character may think that another character is a friend, while this second character may think the first one is a foe.

There are two mechanisms that make the affinity values change. The first one is the lack of interaction between two characters; in this case, the affinity values move toward the indifferent level. The second one is through interactions among characters, which obey a few simple rules. Each affinity level encompasses a set of possible interactions, so when dealing with another character, only actions pertaining to the affinity level between the two can be used. Otherwise, characters ignore interactions that are not contained in their perceived affinity level, and receiving such proposals penalizes the affinity with the character proposing them. Foes constitute an exception to this rule, as they carry out whatever they intend to do irrespective of the other character's intentions.

When receiving a proposal to do something together, a character may decide to either accept or reject it. If the proposal is accepted, both characters increase their mutual affinity values. If it is rejected, the proposer penalizes its affinity with the receiver. Actions for the same level of affinity have a different impact on it. For example, a romantic dinner has a stronger influence on affinity than going to the cinema together. Similarly, the negative effect of rejecting an invitation is opposite to the positive effect of accepting it. Two sample evolutions of the affinities between characters can be seen in **Figure 2**.

Stories as narrations of observed facts

It is important to keep in mind that stories, regardless of the specific features that we have come to associate with them

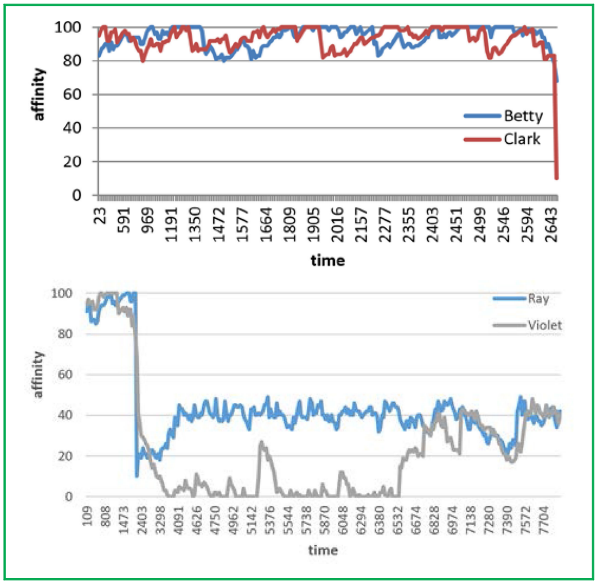


Figure 2

Evolution of the affinities between different characters in the Charade system.

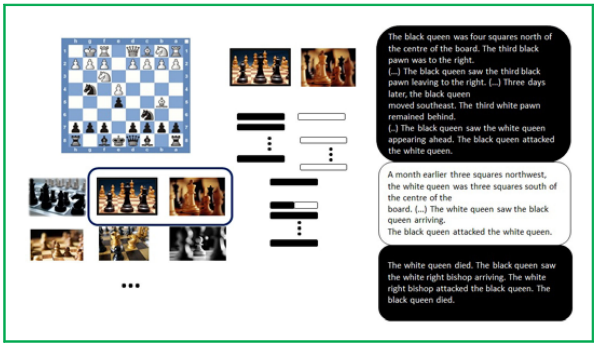


Figure 3

Operation of the Raconteur system. The description of a chess game from a file in an algebraic notation is parsed into a set of narrative threads for individual pieces—each covering a limited range of perception around the piece. A subset of these narrative threads is selected as focalized views of the story— the two queens in this example. The chosen threads are spliced into a single discourse, adding statements to contextualize shifts in time and/or space as needed.

as value-added factors, originate essentially as tools for communicating a given state of the world or the changes that affect it. The systems described in this section address this specific role of stories as vehicles for communicating a particular take on observed reality.

When the task of a storytelling author is considered, there are several possible ways of viewing it depending on what the sources and the purpose of the intended story are. Although storytelling systems in the past had considered mostly the task of generating fictional stories, the systems described in this section consider the generation of stories either to convey facts observed from the real world or to generate stories inspired by facts of this type.

The Raconteur system composes discourses to communicate (a selection of) the set of facts in a chess game [26]. The composition process operates as a self-evaluating cycle, in which the discourse that has been constructed at each point is decoded into a description of the facts it should communicate, and its quality measured in terms of how the interpreted version of the facts compares with the original. The operation of the system, which involves phrasing the events to be considered from the point of view of a participating agent (narrative thread centered on a particular character), selecting a set of these, and then splicing them into a single discourse, is described in **Figure 3.**

This self-evaluating cycle can be understood as a baseline implementation of the reviewing stage of the writing task, as described by Flower and Hayes [27]. It can also be considered as an implemented instance of the Interpretation, Composition, Telling, Interpretation and Validation of Stories (ICTIVS) model described later in this paper.

The StoryFire system [28, 29] models the process of "fictionalizing" real-life events, whereby a set of facts that inspire a story, but either lacks the structure or the clear motivation for the characters that one would expect in a story, is enriched or adapted until the facts coalesce into a surface form that exhibits the properties that make humans consider them a valid story. The computational model of the task of storifying a set of observed events involves: identification of a focalized narrative thread involving the events, as in the Raconteur system; selecting a plot structure that partially matches the chosen thread; mapping characters in the thread to characters in the plot; and generating a readable version of the resulting discourse. An example of a story generated in this way from a selected subset of moves of a chess game is given in **Figure 4.**

Stories as suspense-driven entertainment

A key feature of stories in our society is that they are produced not only for communication but also for entertainment. The entertainment value of stories is often determined by the suspense that they induce on the reader. This section describes a system developed to model this characteristic.

Delatorre et al. [30] propose the architecture of a system whose main objective is the adaptation of the descriptive elements of a scene in such a way that the amount of information of the scene output is adjusted to the required suspense intensity. This architecture is represented in **Figure 5.**

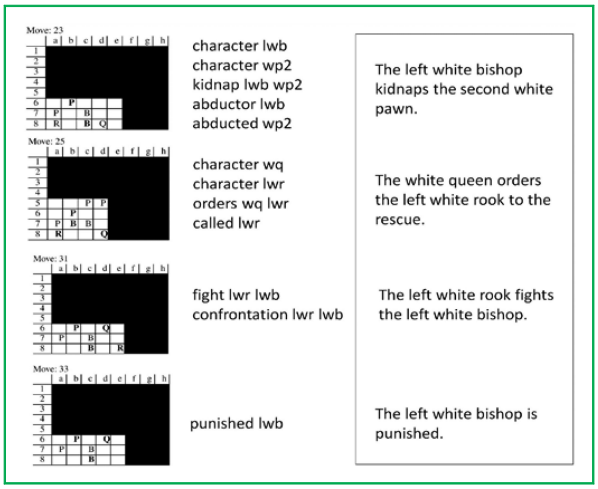


Figure 4

Example of a story produced by the StoryFire system. The narrative threads obtained for a given chess game by the Raconteur system are matched with a set of plots—mined from Booker [34]—based on the presence of relevant characters as given for a particular mapping between chess pieces present at each point in the thread and narrative roles to be instantiated at each point in the plot. For a given alignment between thread and plot, a rendering as a story is produced relying on Raconteur functionality.

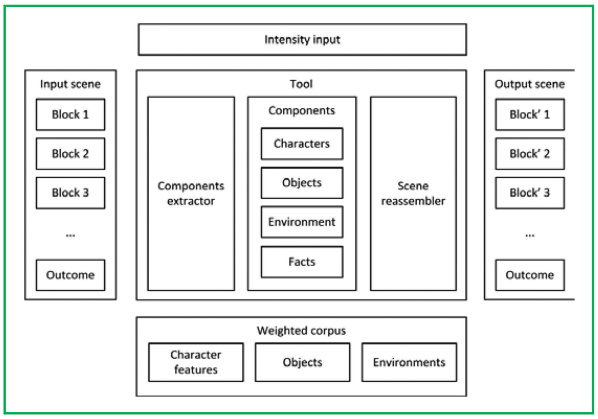


Figure 5

Architecture for rewriting scenes to maximize suspense.

The system receives as input a given scene—represented in terms of a sequence of blocks followed by an outcome—and a number indicating the desired intensity of suspense. From the input scene, it extracts a number of components—characters, objects, environment, and facts—that it then analyzes based on a weighted corpus in which each concept is associated with a quantitative value that represents its level of suspense. Examples of relevant features to establish this value have been shown to be: for characters, empathy and relative strength of individuals in conflict, or for objects, their functional nature that might affect the outcome—as in weapons or possible exits. The values were empirically obtained from experiments where human participants were asked to rate variations of a given story controlled for contrasting instantiations of the features [31]. Based on a computed overall value for suspense at each point arising from the various concepts present, the system then reassembles the scene—replacing concepts with differently valued alternatives—in order to achieve the desired level of suspense.

Relation to prior classifications

Each of the systems described so far presents characteristics that would classify it differently along the taxonomies established by prior researchers, which were reviewed in the section *Existing approaches to narrative generation.*

The PropperWryter system might be classified along the lines of Bailey's story models, Niehaus's systems that decide based on narrative elements, or Kybartas and Bidarra systems focusing on plot.

In contrast, the essential focus on simulation followed by the STellA system would place it more along Bailey's world models, Niehaus's world simulators, or Kybartas and Bidarra systems focusing on place. Yet in this case, the use of narrative curves as a decision mechanism includes elements that would fall under Bailey's story models, or Kybartas and Bidarra systems focusing on plot.

The Charade system relies on world models at a different level of representation, more focused on characters, and in that sense would fall under Bailey's world models and Kybartas and Bidarra systems focusing on place.

The Raconteur and StoryFire systems have a fundamental ingredient of modeling the author's task, which would classify them as Bailey's author models, focused on a very specific version of the storytelling task that addresses the distinction between factual and fictional stories. However, the fact that they start from input in terms of a world model would mean that they also be considered as a reader model. In a sense, this system establishes a bridge between world models and story models, in as much as it encodes a procedure for connecting a view of the world with a representation of it as a story, and one that takes a model of the reader into account to inform the process.

The suspense-driven scene rewriter system is also addressing Bailey's author models in the sense that it focuses on generating suspense for entertainment purposes. However, it also classifies as a reader model in the sense that its whole goal is to produce a particular effect on the reader.

Integrating subtasks into an account of storytelling

It is clear that the different systems described so far address features of stories that are relevant to their perceived quality, and that they emulate different functionalities that human storytellers bring to the task. Each of the systems constitutes a first approximation at modeling how that particular functionality contributes to storytelling. This poses two interesting questions. First, further understanding of how these functionalities cooperate together to the generation of stories is required. Second, some means for combining together the existing computational solutions for these subtasks would reduce the cost of developing solutions.

Key processes in narrative communication

The field of storytelling would benefit greatly from a theoretical model of the task as a composition of simpler processes. Such a model might propose protocols or procedures for how the various functionalities outlined above contribute to one another and to the overall shape of a story. We have attempted to put together a pragmatic description of how some of the tasks that had been identified as part of our research fit in an operational model that articulates them into a computational process. This model is not intended as a description of how humans address the storytelling task, but rather as a set of interactions between subtasks that might replicate some of the behavior observed in humans.

The different kinds of communication between people that can be considered to have a "narrative" quality involve a complex feedback cycle that may include tasks for inventing content, organizing content, interpreting content, and validating content. Instances of this type of cycle may include all possible tasks (when an author invents a plot, revises it, develops the discourse to tell it, then revises that) or only some of them (when someone processes the memory of the day at work into a story to tell the family on arriving home—which is unlikely to involve any invention of content, but may include careful consideration of what they are going to infer from what is told—or when a child invents a story as he plays—which is unlikely to include revision of any kind). A theoretical description of this set of tasks has been postulated in the ICTIVS model [13].

In order to capture separately the tasks of coming up with the content to be transmitted and encoding it into the discourse that will form the message, and to consider the process of tentative interpretation implied in its validation, the model includes five stages: *INVENTION*—creating or establishing the content for the message; *COMPOSITION*—constructing the discourse that conveys the message; *INTERPRETATION*—applying mirror processes to those that will be applied by the receiver to estimate what will be understood; *VALIDATION*—predicting the impact that the message as sent, via the interpreted constructed from it, may have on the receiver; and *TRANSMISSION*—actual operation of sending the message to the receiver. The sender may cycle over the first four stages until satisfied that the interpretation and the impact predicted by her own mirror processes correspond to the reaction she expects from the receiver, and only then will she decide to transmit the message. This potential cycle through the stages is represented in **Figure 6.**

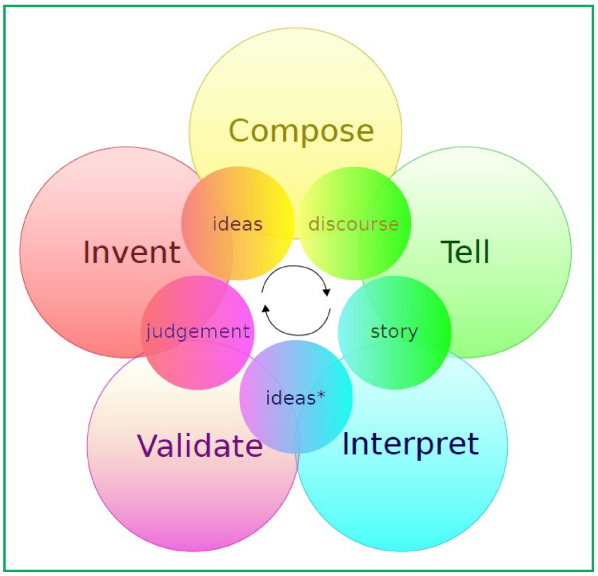


Figure 6

Illustration showing the potential cycle through the stages of the ICTIVS model. In particular cases, the task may take place involving only some of the stages.

Microservice architecture for combining computational storytelling subtasks

The existence within the group of several systems implementing different computational solutions for the storytelling tasks and the insights described above concerning the need to find ways of integrating the different tasks into a single operational process lead to attempts to provide efficient computational solutions for combining together different storytelling functionalities.

Afanasyev [32] is a microservice-based architectural framework for enabling the construction of distributed story generation systems. The core concept behind its model is allowing the existing story generation systems to break their functions into independent services and integrate them in a collaborative environment for generating stories. The main advantage of this approach is the composition of diverse generation models through easily replaceable microservices.

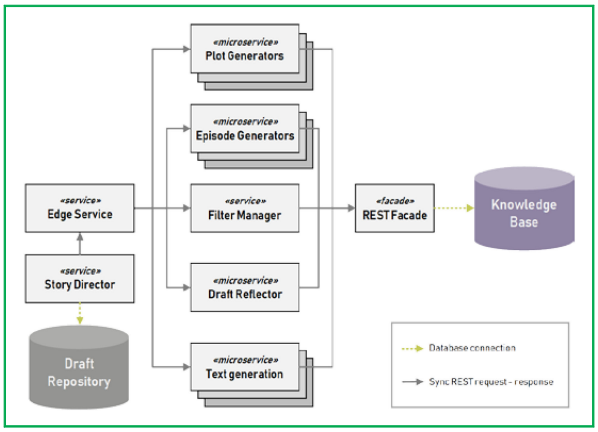


Figure 7

Graphical depiction of the Afanasyev's reference architecture.

Promoting the interoperability between such diverse components entails the development of an associated shared representation model [33]. This model not only considers a mere syntax formalism, but also a knowledge representation model as a base for a collaborative environment to run an enhanced process of literary creation. This representation model requires a knowledge base that can be fed with data from the diverse storytelling systems.

Also, the addition of a preexisting service usually entails the development of an Afansyev's signature-compliant wrapper, due to the need of adapting it to the framework-specific representation model.

Every service of the reference architecture fits a predefined role, as depicted in **Figure 7**. Every role is formalized by a REST-based signature that must be implemented by the microservice that carries it into execution.

The component that drives the whole ecosystem is the Story Director. Its primary goal is to orchestrate the activity of the different microservices when performing the story generation process. To this end, it invokes systematically the REST APIs defined in the framework and provided by the microservices. The behavior of the Story Director is provided according to the Strategy pattern approach, so it can be easily replaced by a different model, and thus the generation process.

The Plot Generator in Afanasyev is responsible for creating the basic structure of a story, namely the plot. It creates a skeleton containing the relevant episodes that happen during the story and the preconditions and postconditions related to every episode. Afanasyev's model allows for maintaining several plot generators that will be conveniently selected by the Story Director to create the plot of a story.

The Episode Generator is the microservice that works in the completion of every episode defined in the plot. It tries to fulfill the preconditions and postconditions established for the episode.

The Filter Manager applies a sequence of predefined filters over every episode for guaranteeing the quality of the generated story in terms of drama, tension, suspense, or any other interesting parameter. If the episode is rejected, then the Story Director requests the Episode Generator to generate a new one.

The Draft Reflector reviews globally the ongoing draft, that is, the story that is being created, and determines if it has been concluded and must be considered a story, or if it requires more iterations for completing the episodes that have not yet been developed in the draft.

Conclusion

The attempts to develop computational models of storytelling have shown that, far from being a well-defined task with clearly specified inputs and outputs, what people understand by the concept of storytelling is in fact a set of considerably diverse operations that are sometimes carried out in isolation to achieve simple stories or specific ingredients that might be a part of stories, and sometimes combined into the production of more elaborate stories.

The systems presented in this paper constitute a small set of examples of computational implementations of instances of these operations. They are not exhaustive in their coverage of processes that might be involved in storytelling. Indeed, there surely exist many others that are equally fundamental—or even more so—and which have yet to be modeled.

The main insight arising from the analysis of this collection of systems is that research efforts in computational storytelling should stop describing themselves under broad and vague labels, and should start being more specific as to which particular feature of stories they focus on, what purpose they build their stories for, and what sources of material they are concerned with including in their stories.

This paper outlines a number of possible descriptions that constitute a first approximation of a vocabulary of constituent tasks of the general storytelling ability. This vocabulary is very much in need of further enrichment if it is to cover the fascinating complexity of the task as carried out by humans.

Another important insight is that models are required of how these constituent tasks of storytelling interact with one another to give rise to the rich stories and complex storytelling processes that we see in human authors. The paper describes two working models—one theoretical and one technological—as initial approximations of how to solve this problem. Nevertheless, this is clearly an open problem that will need significant further work.

Acknowledgment

This work was supported in part by the Spanish Ministry of Economy, Industry and Competitiveness under Project IDiLyCo (TIN2015-66655-R), and in part by the Universidad Complutense de Madrid under Project FEI INVITAR-IA.

References

1. J. Bruner, "The narrative construction of reality," *Crit. Inquiry*, vol. 18, no. 1, pp. 1–21, 1991.
2. C. Salmon, *Storytelling: La Machine à Fabriquer Des Histoires et à Formater Les Esprits*. Paris, France: Découverte, 2007.
3. P Gervás, "Computational approaches to storytelling and creativity," *AI Mag.*, vol. 30, no 3, pp. 49–62, 2009.
4. B. Kybartas and R. Bidarra, "A survey on story generation techniques for authoring computational narratives," *IEEE Trans. Comput. Intell. AI Games*, vol. 9, no. 3, pp. 239–253, Sep. 2017.
5. S. Colton and G. Wiggins, "Computational creativity: The final frontier?" in *Proc. 20th Eur. Conf. Artif. Intell.*, 2012, pp. 21–26.
6. M. O. Riedl and R. M. Young, "Narrative planning: Balancing plot and character," *J. Artif. Intell. Res.*, vol. 39, pp. 217–268, 2010.
7. S. Turner, "MINSTREL: A computer model of creativity and storytelling," *Ph.D. dissertation*, Univ. California at Los Angeles, Los Angeles, CA, USA, 1992.
8. M. Theune, E. Faas, A. Nijholt, et al., "The virtual storyteller: Story creation by intelligent agents," in *Proc. Technol. Interact. Digit. Storytelling Entertainment*, 2003, pp. 204–215.
9. P. Bailey, "*Searching for storiness: Story-generation from a reader's perspective*," in *Narrative Intelligence*. Menlo Park, CA, USA: AAAI Press, pp. 157–163.1999.
10. P. Gervás, B. Lönneker-Rodman, J. C. Meister, et al., "Narrative models: Narratology meets artificial intelligence," in *Proc. Int. Conf. Lang. Resour. Eval., Satellite Workshop, Toward Comput. Models Literary Anal.*, Genova, Italy, 2006, pp. 44–51.
11. B. O'Neil, "A computational model of suspense for the augmentation of intelligent story generation," Ph.D. dissertation, Georgia Inst. Technol., Atlanta, GA, USA, 2013.
12. J. Niehaus, "Cognitive models of discourse comprehension for narrative generation," Ph.D. dissertation, North Carolina State Univ., Raleigh, NC, USA, 2011.
13. P. Gervás and C. León, "The need for multi-aspectual representation of narratives in modelling their creative process," in *Proc. Workshop Comput. Models Narrative*, Canada, 2014, pp. 61–76.
14. P. Gervás, "Computational drafting of plot structures for Russian folk tales," *Cogn. Comput.*, vol. 8, pp. 187–203, 2015.
15. V. Propp, *Morphology of the Folktale*. Austin, TX, USA: Univ. Texas Press, 1968.
16. S. Colton, Llano, M. T., Hepworth, R., et al., "The beyond the fence musical and computer says show documentary," in *Proc. 7th Int. Conf. Comput. Creativity*, Paris, France, 2016, pp. 311–320.
17. C. León and P. Gervás, "Creativity in story generation from the ground up: Non-deterministic simulation driven by narrative," in *Proc. 5th Int. Conf. Comput. Creativity*, Ljubljana, Slovenia, 2014, pp. 201–210.
18. C. León and P. Gervás, "Prototyping the use of plot curves to guide story generation," in *Proc. Workshop Comput. Models Narrative, Lang. Resour. Eval. Conf.*, 2012, pp. 152–156.
19. G. Méndez, P. Gervás, and C. León, "On the use of character affinities for story plot generation," in *Knowledge, Information and Creativity Support Systems: Selected Papers From KICSS'2014*, vol. 416. Berlin, Germany: Springer, 2016, pp. 211–225.
20. J. W. von Goethe, *Elective Affinities / Kindred by Choice*. New York, NY, USA: Holt & Williams, 1809.
21. G. Polti, *The Thirty-Six Dramatic Situations*, The Editor Company, Ridgewood, New Jersey, USA. 1917.
22. M. Si, S. C. Marsella, and D. V. Pynadath, "Thespian: Modeling socially normative behavior in a decision-theoretic framework," in *Intelligent Virtual Agents, Lecture Notes in Computer Science*, vol. 4133. Berlin, Germany: Springer, pp. 369–382, 2006.
23. W. L. Johnson and A. Valente, "Tactical language and culture training systems: Using artificial intelligence to teach foreign languages and cultures," in *Proc. 20th Nat. Conf. Innov. Appl. Artif. Intell.*, 2008, pp. 1632–1639.
24. M. S. El-Nasr, J. Yen, and T. R. Ioerger, "FLAME–fuzzy logic adaptive model of emotions," *Auton. Agents Multi-Agent Syst.*, vol. 3, no. 3, pp. 219–257, 2000.
25. R. Imbert and A. de Antonio, "An emotional architecture for virtual characters," in *Proc. Int. Conf. Virtual Storytelling*, 2005, pp. 63–72.
26. P. Gervás, "Composing narrative discourse for stories of many characters: A case study over a chess game," *Literary Linguist. Comput.*, vol. 29, pp. 511–531, 2014.
27. L. Flower and J. R. Hayes, "A cognitive process theory of writing," *Coll. Composition Commun.*, vol. 32, no. 4, pp. 365–387, 1981.
28. P. Gervás, "Storifying observed events: Could I dress this up as a story?" in *Proc. 5th AISB Symp. Comput. Creativity*, Liverpool, U.K., paper 5, 2018.
29. P. Gervás, "Targeted storyfying: Creating stories about particular events," in *Proc. 9th Int. Conf. Comput. Creativity*, Salamanca, Spain, 2018, pp. 232–239.
30. P. Delatorre, B. Arfè, P. Gervás, et al., "A component-based architecture for suspense modelling," in *Proc. AISB's 3rd Int. Symp. Comput. Creativity*, 2016, pp. 32–39.
31. P. Delatorre, C. León, P. Gervás, et al., "A computational model of the cognitive impact of decorative elements on the perception of suspense," *Connection Sci.*, vol. 29, no. 4, pp. 295–331, 2017.
32. E. Concepcion, P. Gervas, and G. Mendez, "Afanasyev: A collaborative architectural model for automatic story generation," in *Proc. 5th AISB Symp. Comput. Creativity*, paper 2, 2018.
33. E. Concepcion, P. Gervas, and G. Mendez, "A common model for representing stories in automatic storytelling," in *Proc. 6th Int. Workshop Comput. Creativity, Concept Inven., Gen. Intell.*, paper 8, 2017.
34. C Booker, *The Seven Basic Plots: Why We Tell Stories*. London, U.K.: Bloomsbury, 2004.

Received February 8, 2018; accepted for publication December 21, 2018

Pablo Gervás *Facultad de Informática, Universidad Complutense de Madrid, Madrid 28040, Spain (pgervas@ucm.es)*. Dr. Gervás is an Associate Professor with the Department of Software Engineering and Artificial Intelligence, School of Informatics, Universidad Complutense de Madrid, Madrid, Spain. He is the Director of the NIL Research Group (Natural Interaction based on Language). Over the years, his research interests have shifted toward studying the role of narrative in human communication, with a view to applying it in human–computer interaction. Besides his generic interest in understanding and modeling language, he tries to tackle computational models of human creativity. His interests include the following lines of research: natural language generation, natural language analysis, NLP for accessibility, and NLP and literary artifacts.

Eugenio Concepción *Universidad Complutense de Madrid, Madrid 28040, Spain (econcepc@ucm.es).* Mr. Concepción received a B.S. degree in computer engineering from Universidad Complutense de Madrid (UCM), Madrid, Spain, and a M.S. degree from Universitat Oberta de Catalunya, Barcelona, Spain. He is currently working toward a Ph.D. degree in computer science at UCM, focused in automatic story generation. He combines this research activity with his responsibilities as CTO in a software development company.

Carlos León *Facultad de Informática, Universidad Complutense de Madrid, Madrid 28040, Spain (cleon@ucm.es)*. Dr. León received a Ph.D. degree in computer science from the Universidad Complutense de Madrid, Madrid, Spain, in 2010. He is currently an Assistant Professor with the Department of Software Engineering and Artificial Intelligence, Complutense University of Madrid, Madrid. His research focuses on computational models of creative behavior, with particular focus on cognitive descriptions of narrative generation. He studies the role of narrative as a fundamental component of human cognition.

Gonzalo Méndez *Facultad de Informática, Universidad Complutense de Madrid, Madrid 28040, Spain (gmendez@fdi.ucm.es)*. Dr. Méndez received a Ph.D. degree in computer science from the Universidad Politécnica de Madrid, Madrid, Spain, in 2008. He is currently an Associate Professor with the Department of Software Engineering and Artificial Intelligence and the Director with the Instituto de Tecnología del Conocimiento, Universidad Complutense de Madrid, Madrid. His current research interests include narrative generation, natural language generation for figurative language, accessibility systems, and agent-based simulations.

Pablo Delatorre *Facultad de Informática, Universidad de Cádiz, Cádiz 11001, Spain (pablo.delatorre@uca.es)*. Prof. Delatorre received B.S. and M.S. degrees in computer engineering from Universidad de Sevilla, Sevilla, Spain, and a Ph.D. degree in computer science at Universidad Complutense de Madrid, Madrid, Spain, in 2018, focused in narrative and interactive suspense applied to automatic story generation. He combines this research activity with his responsibilities as a Teacher with the University of Cadiz, Cadiz, Spain. Outside the academic profession, he worked ten years in the ICT sector, managing national and international corporate projects.

Capítulo 5

Técnicas de generación de historias con múltiples tramas

"Plot is no more than footprints left in the snow after your characters have run by on their way to incredible destinations"
— Ray Bradbury, Zen in the Art of Writing

5.1. Introducción al proceso de generación de tramas

En esta sección se presenta el trabajo realizado en materia de generación de historias con múltiples tramas. Las técnicas principales, recogidas en los artículos contenidos en el capítulo, se resumen a continuación con vistas a destacar las diferencias entre ellas y la forma en que generan resultados.

La generación de una historia con múltiples tramas requiere que estas se ajusten a ciertos parámetros comunes. Por ejemplo, todas las tramas a combinar deben compartir la misma ambientación y un subconjunto de la unión de todos los personajes participantes en cada una de las tramas individuales.

El proceso de generación de las tramas se basa en (Concepción et al., 2019) y (Concepción et al., 2020). Se emplea un motor basado en plantillas que produce las tramas de una historia. Dichas plantillas son un subconjunto de las tramas básicas cinematográficas compiladas por Balló y Pérez (2007). Las plantillas están almacenadas en un repositorio donde tienen asociados metadatos que permiten obtener la plantilla o plantillas que se ajusten a determinados parámetros de búsqueda.

5.2. Técnicas de entrelazado de tramas

La construcción de historias multi-trama puede lograrse combinando dos o más tramas sencillas como subtramas de una historia mayor. Para simplificar, los ejemplos de este documento sólo consideran combinaciones de tramas simples que consisten en instancias de una sola plantilla de trama, pero en términos generales, las tramas que se combinan pueden ser en sí mismas complejas.

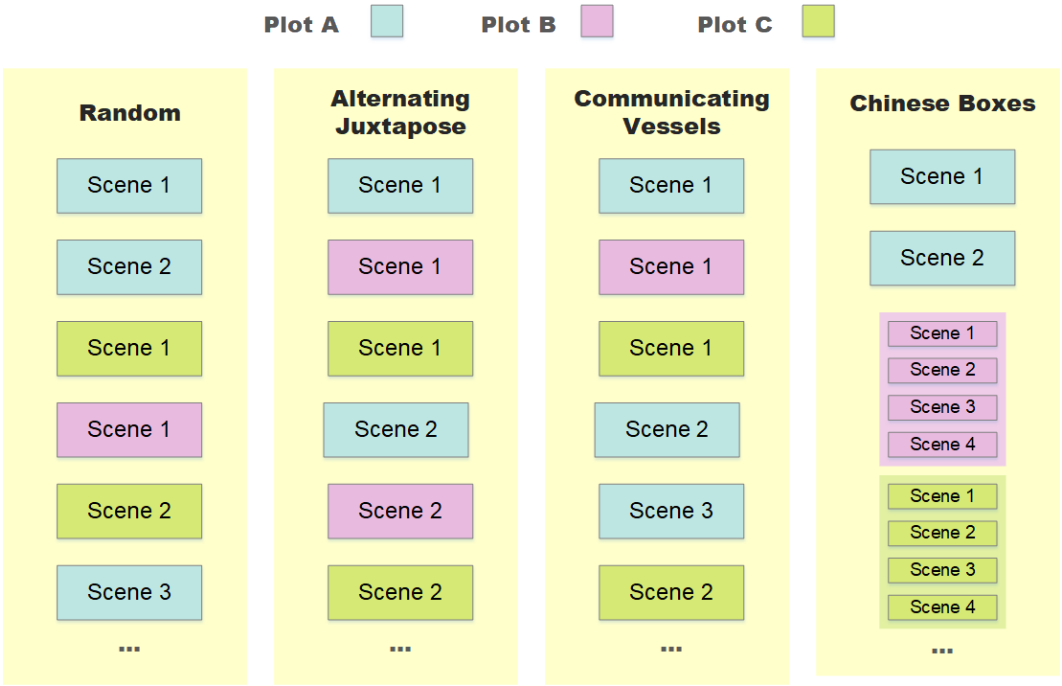


Figura 5.1: Resumen visual de las diversas técnicas de entrelazado de tramas

Cuando se desea generar una historia con varias tramas, el director de historias realiza varias peticiones al generador de tramas para que cree las subtramas conforme a ciertos parámetros comunes (por ejemplo, ambientación, personajes compartidos, etc.). Para cada subtrama puede seleccionar una plantilla distinta. En cada petición, el generador de tramas genera una subtrama instanciando la plantilla y asocia los roles de la plantilla con los personajes que participan en dicha subtrama. La subtrama generada es una lista de escenas a modo de borrador. La plantilla determina cuántas escenas debe tener la subtrama y qué personajes actúan en cada escena, de acuerdo con la función que tienen asignada en la plantilla, pero el contenido de la escena está aún sin definir. Este proceso de instanciación de personajes se basa en el conocimiento del contexto en el que se desarrollará la historia. Cuando el Director de Historias considere que el número de subtramas generadas es el adecuado, entonces pasa una petición al tejedor de tramas para que genere una trama unificada a partir del conjunto de subtramas generadas aplicando la técnica de entretejido concreta.

El proceso de entrelazado de tramas se basa en la aplicación de dos técnicas: la técnica de los *Vasos comunicantes* y la técnica de las *Cajas chinas*, ambas estrategias están inspiradas en el trabajo de Menéndez (2013). La implementación de dichas técnicas se recoge con detalle en *Evolving the INES story generation system: from single to multiple plot lines* (Concepción et al., 2019) y especialmente en *Exploring Baselines for Combining Full Plots into Multiple-plot Stories* (Concepción et al., 2020), donde se presentan diversos ejemplos y se realiza un análisis cualitativo de las historias multitrama generadas con las mismas.

Ambas se comparan entre sí en la figura 5.1.

5.2.1. Técnicas base: aleatoriedad y alternancia

Como se muestra en la figura 5.1, además de las técnicas principales, en la investigación se consideraron dos técnicas que podrían llamarse base, puesto que servían para comparar la efectividad de las principales, tal como se recoge en el artículo *Exploring Baselines for Combining Full Plots into Multiple-plot Stories* (Concepción et al., 2020). Dichas técnicas son el entrelazado, o entretejido, aleatorio y la yuxtaposición alternante. En el primer caso, la integración de las escenas de las diferentes subtramas se realiza de forma aleatoria, sin ningún tipo de verificación de consistencia. En el segundo caso, la integración se realiza alternando en orden una escena de cada subtrama, sin realizar tampoco ningún chequeo de consistencia. Los resultados generados con estas dos aproximaciones permiten realizar comparaciones cualitativas entre los resultados obtenidos con las dos técnicas principales, tal como se expone en el citado artículo (Concepción et al., 2020).

5.2.2. Técnica de los vasos comunicantes

La técnica de los *Vasos Comunicantes* (Menéndez, 2013) se basa en construir una historia alternando al menos dos líneas argumentales paralelas diferenciadas –ver figura 5.1. Un ejemplo de esta técnica puede verse en "Madame Bovary" (Flaubert, 1857), que contiene un capítulo que alterna dos líneas argumentales aparentemente desconectadas. El único punto en común entre ellas es el contexto temporal en el que ocurren las acciones y los sucesos que contienen. El efecto resultante es una "contaminación" entre las dos líneas argumentales que, tomadas aisladamente, producirían una comprensión diferente en el lector. En otras palabras, cuando dos pasajes argumentales separados y alternantes se entrelazan, la proximidad y la alternancia generan una influencia mutua. Esta influencia puede aplicarse al tono, la tensión o la atmósfera que cada historia transmite a la otra. Este efecto se produce principalmente en el proceso de interpretación que lleva a cabo el lector.

En el algoritmo de entrelazado por vasos comunicantes las subtramas a combinar se procesan secuencialmente en un ciclo, añadiendo en cada punto al borrador una escena de la siguiente subtrama si es compatible con el borrador hasta el momento en términos de satisfacción de la precondición. Se da prioridad a la inserción de escenas de una subtrama diferente, pero si no se encuentra ninguna opción consistente, se añade la siguiente escena de la misma subtrama.

Se puede expresasr más formalmente de la siguiente manera: Dado un índice j, tal que 1 <j $\leq$ n, un conjunto de tramas P_1, ...P_n donde n>1 y donde cada trama P_i está formada por una secuencia de escenas s_i^1, ...s_i^m, siendo m>1; entonces, la heurística de los vasos comunicantes seleccionará la siguiente escena de la trama combinada efectuando una búsqueda lineal hasta encontrar una escena s_i^k tal que precondiciones(s_i^k) son consistentes con postcondiciones(s_j^l), siendo s_j^l la última escena insertada en la trama combinada. En el peor de los casos, $s_i^k = s_j^{l+1}$, es decir la siguiente escena dentro de la misma trama.

5.2.3. Técnica de las cajas chinas

La técnica del *Frame Story*, o la técnica *cajas chinas*, como la denomina Menéndez (2013), se basa en la inclusión de historias anidadas dentro de una historia más larga. Cada historia anidada puede estar relacionada con uno o varios personajes de la línea argumental principal, o incluso consistir en una historia separada, creada con el propósito de explicar algunos sucesos de la historia principal. La figura 5.1 muestra un ejemplo visual del anidamiento que produce la aplicación de esta técnica.

Esta es una técnica literaria bastante común que se ha aplicado en grandes obras de la literatura universal como *"Las mil y una noches"* (Vernet, 1990) y *"Don Quijote de la Mancha"* (de Cervantes Saavedra, 1605). El objetivo del relato secundario es servir de pieza de acompañamiento a otro relato, dentro de éste, donde una narración introductoria o principal prepara el terreno para una segunda narración más acentuada o para un conjunto de relatos más cortos. Esta disposición conduce la lectura desde un primer relato a uno o varios relatos dentro de éste. También puede servir para informar a los lectores sobre aspectos clave de la narración que, de otro modo, serían difíciles de entender.

En este algoritmo se selecciona inicialmente una trama como trama principal y se añade al borrador en curso. El resto de tramas que se combinan se procesan secuencialmente. Para cada trama adicional, el algoritmo intenta encajarla entre dos escenas del borrador, basándose en si las precondiciones de la primera escena y las postcondiciones de la última escena de la trama a insertar son compatibles con las postcondiciones de la escena anterior y las precondiciones de la siguiente escena del borrador, respectivamente. Esencialmente, comprueba si el hueco entre dos escenas del borrador es adecuado para insertar una trama secundaria completa. Esta técnica puede dar lugar a cualquier número de niveles de anidamiento de las tramas secundarias de una historia, restringido únicamente por el número de tramas secundarias disponibles (y las restricciones de satisfacción de las precondiciones).

Se puede expresar de la siguiente forma: Dado un índice j, tal que 1 <j $\leq$ n; una trama principal P_0; un conjunto de tramas P_1, ...P_n donde n>1 y donde cada trama P_i está formada por una secuencia de escenas s_i^1, ...s_i^m, siendo m>1; entonces, la heurística de los vasos comunicantes seleccionará la siguiente escena de la trama combinada efectuando una búsqueda lineal hasta encontrar una escena s_i^1 tal que precondiciones(s_i^1) son consistentes con postcondiciones(s_0^j). En caso de no encontrar una escena compatible, j se incrementará en 1 y se repetirá el proceso, hasta llegar a la última escena de P_0.

5.2.4. Aproximación evolutiva a la generación de historias con varias tramas

En una situación en la que se conoce previamente cuáles son las subtramas que se pretende combinar, la tarea de encontrar una combinación óptima de estas subtramas requiere explorar un espacio de búsqueda complejo. Dicho espacio de búsqueda debe considerar combinaciones de las escenas de las distintas subtramas en un número elevado de posibles órdenes cronológicos relativos, y considerando en cada caso una variedad grande de alineaciones entre personajes específicos de cada subtrama.

Para explorar este espacio de búsqueda tan grade pueden resultar útiles las soluciones evolutivas (Yar et al., 2019). En estas soluciones, se trabaja con una población de borradores que sean posibles respuestas al problema. Cada borrador tiene una representación genética que determina la solución al problema. Para encontrar una respuesta, se genera una sucesión de poblaciones alternativas (generaciones) a partir de la población anterior a base de mutar la representación genética de cada borrador o cruzarla con la de otro borrador. Los individuos que sobreviven para pasar a la siguiente generación son los aquellos a los que una función de evaluación para el problema asigna un valor mayor que los competidores.

Para aplicar estas técnicas a un problema de generación de historias con múltiples tramas es necesario:

- una representación genética que capture tanto la alternancia entre escenas de las distintas subtramas como las potenciales alineaciones entre personajes de distintas subtramas
- mecanismos de mutación y cruce que permitan explorar el espacio completo de órdenes relativos entre escenas de distintas tramas y potenciales alineaciones entre personajes de distintas subtramas
- una función de evaluación que asigne puntuaciones positivas a los individuos (borradores) que presenten combinaciones de tramas válidas y puntuaciones negativas a los individuos que correspondan a combinaciones problemáticas

Este es el tipo de solución que se explora en el artículo "Evolutionary Construction of Stories that Combine Several Plot Lines" (Gervás et al., 2022). La representación genética propuesta recurre a dos soluciones distintas, cada una dedicada a codificar uno de los aspectos relevantes (orden relativo de las escenas, alineaciones entre personajes). Esta representación genética está diseñada de modo que operadores clásicos de mutación y cruce sean aplicables. La función de evaluación utilizada está diseñada a partir de las métricas de calidad de historias planteadas en el artículo "Assessing MultiPlot Stories: from Formative Analysis to Computational Metrics" (Gervás et al., 2021), que a su vez son el resultado de la evaluación formativa llevada a cabo sobre los ejemplos de historias descritos en el artículo "Exploring Baselines for Combining Full Plots into Multiple-plot Stories" (Concepción et al., 2020).

Los detalles de los procesos concretos en cada caso se pueden encontrar en los artículos.

5.3. Artículo: Evolving the INES story generation system: from single to multiple plot lines

Este artículo resume los trabajos para generar una nueva versión del sistema generador INES, descrito en el Capítulo 4, para que genere historias con múltiples tramas. En este artículo se recogen por primera ver las dos técnicas mencionadas en la sección anterior: los *vasos comunicantes* y las *cajas chinas*. Además, el nuevo INES implementa el servicio *tejedor de tramas*, descrito también en el Capítulo 4

y perteneciente a la arquitectura de referencia de Afanasyev. Esta implementación pone a prueba las dos técnicas indicadas anteriormente.

5.3.1. Cita completa

E. Concepción, Gervás, P., and Méndez, G. (2019) *Evolving the INES story generation system: from single to multiple plot lines*, in 10th International Conference on Computational Creativity (ICCC 2019), UNC Charlotte, North Carolina, USA.

5.3.2. Resumen original

INES (Interactive Narrative Emotional Storyteller) is a story generation system based on the Afanasyev framework. It is focused on generating stories by combining template-based plot generation with an agent-based simulation of characters' interaction. Its design follows the microservice-oriented model established by Afanasyev, in which a Story Director orchestrates the story generation stages, implemented by specialised microservices. While this model is suitable for generating a single plot story, it is insufficient for managing a multiple plots scenario. This paper focuses on describing an evolved version of INES that aims at generating stories that contain different plot lines. In addition to the adoption of changes in the story representation model, the adaptation entails a modification of the operation of INES and includes a new microservice: the Plot Weaver. This component introduces the application of a literary technique referred to as the "Communicating Vessels", in which the different lines evolve in parallel while interacting between themselves.

Evolving the INES story generation system: from single to multiple plot lines

Eugenio Concepción[1] and **Pablo Gervás**[1,2] and **Gonzalo Méndez**[1,2]
[1]Facultad de Informática
[2]Instituto de Tecnología del Conocimiento
Universidad Complutense de Madrid
{econcepc,pgervas,gmendez}@ucm.es

Abstract

INES *(Interactive Narrative Emotional Storyteller)* is a story generation system based on the Afanasyev framework. It is focused on generating stories by combining template-based plot generation with an agent-based simulation of characters' interaction. Its design follows the microservice-oriented model established by Afanasyev, in which a Story Director orchestrates the story generation stages, implemented by specialised microservices. While this model is suitable for generating a single plot story, it is insufficient for managing a multiple plots scenario. This paper focuses on describing an evolved version of INES that aims at generating stories that contain different plot lines. In addition to the adoption of changes in the story representation model, the adaptation entails a modification of the operation of INES and includes a new microservice: the Plot Weaver. This component introduces the application of a literary technique referred to as the *"Communicating Vessels"*, in which the different lines evolve in parallel while interacting between themselves.

Introduction

Automated story generation is a research area in Artificial Intelligence focused on developing systems whose result is a story (Gervás 2012). It is closely related to other Computational Creativity areas such as interactive storytelling (Glassner 2009).

Story generation systems present two distinctive characteristics: they strongly depend on knowledge and they can only generate a constrained variety of stories (Gervás and León 2014). A story generator requires knowledge from practically all areas of the collective wisdom, so it needs to be fed with a wide range of information, from the most basic common sense knowledge to the physical world rules. Besides this, due to the technical limitations of the generation process, story generators only generate stories of a certain kind –in terms of theme, rhythm, discourse, etc. Many of these limitations come from the fact that the architecture of these systems is built according to a monolithic design. Hence, a single application concentrates all the required functionality and assets. If this is combined with the lack of architectural mechanisms for collaborating with other systems, the results obtained are quite restrictive. A way of addressing all these limitations is by adopting a distributed architecture, with an emphasis on the collaboration between different systems.

INES (Concepción, Gervás, and Méndez 2018b) is a story generation system based on Afanasyev, a microservice-oriented architectural framework (Concepción, Gervás, and Méndez 2018a). Afanasyev provides a reference architecture that brings a collaborative environment for services sourced from different storytelling systems and orchestrated by a Story Director. INES was originally developed to test the suitability of the framework and also as an evolution of the Charade storytelling system (Méndez, Gervás, and León 2016). The limitation of the current version of INES comes from the fact that it can only generate a single-plot story, being its design inadequate for managing a multiple plots scenario.

Background

TALE-SPIN (Meehan 1977), one of the first story generators, wrote short stories about the inhabitants of a forest. From a technical point of view, it applied planning techniques (Cohen and Feigenbaum 2014) for generating the characters actions –while trying to achieve their goals, and then it wrote up the story by narrating the steps performed by the characters for achieving their goals. **Author** (Dehn 1981) was also a planner but focused on the authorial goals instead of the characters' goals. **Universe** (Lebowitz 1984) generated scripts for a TV soap opera by focusing on characters interaction. Its generation process included a planning stage that kept track of pending goals for developing a partial draft of the story until plot completion. **GESTER**, *GEnerating STories from Epic Rules*, (Pemberton 1989) was one of the first approaches towards generating stories from interacting modules of independent knowledge. The program was a rule-based story generation system that managed information about story structure, in the form of a simplified version of a narrative grammar, and to the possible events and actors of the epic sub-genre. **Brutus** (Bringsjord and Ferrucci 1999) generated short stories about betrayal. It used a very thorough knowledge model for representing the concept and implication of betrayal. It also provided a grammar-based generation model and a literary beautifier, which allowed it to generate high-quality stories, providing texts that could have been written by humans. **MEXICA** (Perez y Perez 1999) was a storytelling system that generated mythological stories about the Mexicas, the early inhabitants of Mexico. It was the first system that brought the character's

emotions into play in the generation process. **MAKEBELIEVE** (Liu and Singh 2002) generated short fictional stories using common sense knowledge to generate them. It required the user to provide a story about a character as initial seed, for later attempting to continue that story by inferring possible sequences of events that might happen to that character. **TEATRIX** (Machado, Paiva, and Brna 2001; Prada, Machado, and Paiva 2000) was a virtual environment for story creation, designed to help children and teachers to understand the whole process of collaborative story creation. It provides an environment where both drama and story creation are merged into one medium. Architecturally, it is an agent-based system in which each character is performed by an intelligent software agent interacting in the story world. Every character plays a role according to the Propp's folktales model (Propp 1968). **Fabulist** (Riedl and Young 2010) is a whole architecture for automatic story generation and presentation which combines both the author interests and the characters intentionality. **Charade** (Méndez, Gervás, and León 2014; 2016) is a storytelling system focused on the relationships between the characters. By simulating their interactions, it tracks the evolution of their mutual affinities and applies it for generating stories. From an architectural point of view, it is an agent-based architecture implemented in JADE (Java Agent Development Framework) (Bellifemine, Poggi, and Rimassa 1999). Charade aims at implementing an affinity model decoupled from the story domain, that is, the world in which the story takes place or any other context-related attribute of the characters.

Related work

Single-plot stories are not the only model in human-made narrative. There are a good number of stories that contain more than one plot, such as unnatural narratives, a subset of fictional narratives that subvert the physical laws, the logic principles or the standard human limitations (Alber and Heinze 2011). This approach affects not only the facts told in the story, but also its structure: it can contain several interwoven plots, a rupture of the plot natural progression, multiple concurrent plots and many other possibilities. From the beginning of the literature, the unnatural elements have been present in the literary production (Todorov 1975).

In addition to the historical precedents, postmodern literature has adopted much of those unnatural resources, bringing a disruptive narrative design to the stories (Martínez 2011). Contemporary literature is fraught with stories containing several plot lines linked together by means of diverse strategies (Menéndez 2013).

There is not a long record of multiple plot generation and combination in automatic story generation. One of the remarkable works in this respect is the plot weaving algorithm proposed by Fay (Fay 2014). This is a method that takes a set of individual plot threads as input, one for every single character, and generates a new story by tying them. These individual plot threads have been previously extracted from a preexisting story by means of the *Genesis* story understanding system (Winston 2016). This plot weaving algorithm makes sure that characters' plots are compatible and it also takes care of building a consistent timeline for all of the plot elements of the story. This procedure is computationally difficult because it entails selecting the best set of pairings of characters to generic entities to create the best possible combination for the story (Fay 2014).

Porteous et al. (Porteous, Charles, and Cavazza 2016) have developed a remarkable example of multiplot interactive storytelling system. It is focused primarily on facing three challenges: the distribution of the characters across the different subplots, the length of each subplot presentation and the transitions between subplots.

Gervás (Gervás 2018) has recently explored the suitability of combining events from a sequence for generating a plot as a technique for story generation. This approach has partially influenced the solution proposed in this paper.

It is also worth mentioning other efforts in the generation of multiple possibilities in a story, such as the planning approach by Li and Riedl (Li and Riedl 2010) and the *Crystal Island* (Mott, Lee, and Lester 2006) interactive narrative engine. In the first case (Li and Riedl 2010), authors define a plan refinement technique based on partial-order planning that it uses for off-line adaptation of authored narratives with multiple "quests" adapting the plot line to create new plausible sequences of actions. The Crystal Island generation model offers multiple quest subplots that encourage the user goal recognition, combining multiple plot elements to create rich customized stories (Mott, Lee, and Lester 2006). However, whilst these approaches sought to generate multiple quests, they did not provide a procedure to interleave them.

Materials and methods

This section covers the technical basis and the conceptual techniques considered for designing the solution. It firstly provides a review of INES, with emphasis on those aspects that mainly take part in the plot generation step, and some known literary techniques for creating stories based on multiple plots. It also focuses on reviewing the knowledge representation model used by INES, taken from the Afanasyev common knowledge representation structure. Lastly, it describes different literary strategies for writing multiple plot narratives.

The INES story generation system

INES (Interactive Narrative Emotional Storyteller) is based on the Afanasyev framework (Concepción, Gervás, and Méndez 2018a; 2018b). One of the declared objectives of this framework is to facilitate the generation of stories much closer to human-made literature by combining the diverse capabilities of various story generators (Concepción, Gervás, and Méndez 2018a). Afanasyev provides both a microservice-oriented reference architecture and a common knowledge representation model.

The architecture of INES is based on microservices. Figure 1 depicts a high-level view of its original components. The following lines focus on a short review of the relevant aspects of the architecture. A more detailed description can be found in (Concepción, Gervás, and Méndez 2018b).

The operation of this microservices ecosystem is driven by the Story Director. It orchestrates the rest of the ser-

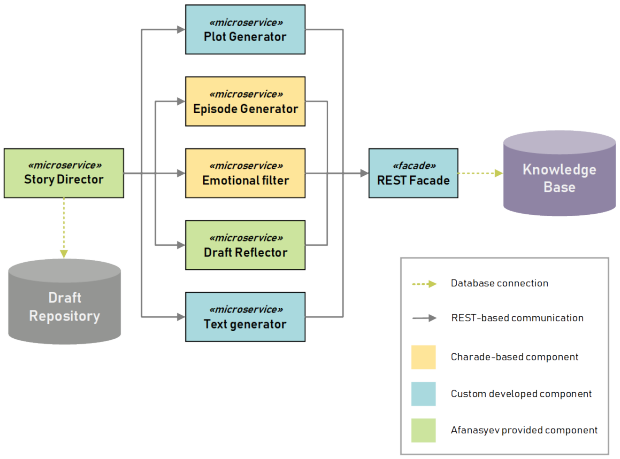


Figure 1: Original architecture of INES.

Episode	Description
Initial state	A peaceful community
Arrival	The arrival of the outsider to the community
Outsider destructive actions	The outsider acts against the members of the community, performing destructive actions, without being uncovered
The outsider revealed	The true evil nature of the outsider is revealed
The rise of the heroes	The The heroes rise from the community and fight against the outsider
Purge	The outsider is purged. The community becomes peaceful again

Table 1: The "Destructive Outsider" story plot template

vices to perform the story generation process. Story Director starts by requesting the Plot Generator to generate a plot. This request includes a list of initial characters, the plot template to apply and the setting in which the story happens. Every plot is a sequence of scenes characterised by a precondition and a postcondition that reflect the state of the world before and after the development of each scene (Concepción, Gervás, and Méndez 2018b; 2018a).

The knowledge representation model provided by Afanasyev tries to cover all the aspects related to the structure and meaning of a story (Concepción, Gervás, and Méndez 2017). This representation has been designed as a hierarchical structure in which the root concept is the **story**. A story represents what both intuitively and narratologically can be considered a story, that is, a narration of events happening in a setting (Concepción, Gervás, and Méndez 2017). It is composed by the two classic narratological components: the plot and the space.

The plot of the story represents its essential structure, providing a sort of scaffolding for the actions and events that happen across the story. In the Afanasyev model, the plot is generated at the beginning by the Plot Generator microservice.

The INES instance for the **Plot Generator** is named **"Audrey"** —after Audrey Hepburn. It is a template-based plot generator which produces outlines from a subset of the cinematographic basic plots compiled by Balló (Balló and Pérez 2007). Audrey aims at building a story plot containing the main scenes that will be completed by the Episode Generator microservice. Its generation model is quite similar to systems like Gester (Pemberton 1989) and Teatrix (Machado, Paiva, and Brna 2001).

The original plot building procedure started by selecting one of the cinematographic templates, namely a conceptual structure with a sketch of the plot, and developed it later by instantiating the roles and the types of actions into real characters and actions. Audrey's REST interface supports the random selection of a template but also the selection of a specific template name. Once a basic template is selected, it instantiates its generic elements to develop a concrete plot. This step entails the use of knowledge about the context in which the story will be set. In this case, the context is inferred from the request parameters.

An example of one of these templates is *"The destructive outsider"*, summarized in table 1 (Concepción, Gervás, and Méndez 2018b).

Audrey queries the knowledge base that contains the main concepts presented in the plot for creating a consistent detail for every episode (Concepción, Gervás, and Méndez 2018b) according to certain setting. In the prior example, the plot mentions a "community", an "outsider", some "destructive actions" performed by the outsider, etc. All these concepts are included in the knowledge base and there are more specific roles and actions which refer to them. For example, the concept of "community" can be instantiated into a "town", a "family" or a "company". In each case, the "outsider" can be a "new sheriff", an "unknown relative" or a "new colleague". By the same token, the knowledge base contains the information required to determine the type of actions that the characters can perform.

Every episode or scene is expressed in terms of a set of attributes that essentially provide information about the story space —including both time and location, the characters that appear in the episode and the state of the world before and after the episode happens (Concepción, Gervás, and Méndez 2018a; 2018b). These last information is represented as the scene precondition and postcondition. They are sets of assertions expressing the restrictions to be considered during the development of the episode's detail. Table 2 shows a sample of all these attributes.

Probably, the most influential attributes in terms of consistency keeping across the scenes are the precondition and the postcondition. They are included as a part of the Scene - Frame - State structure. These attributes are a collection of assertions about the state of the story world before and after the scene occurs. They contain assertions related to any existant referenced in the scene —that is, characters, beings

Attribute	Value
Precondition	John is a friend of William
Postcondition	John is a friend of William John performs friendly actions to William
Characters	John, William
Time	Story relative time in which the episode happens
Location	Spatial reference in the story world

Table 2: The basic attributes of an episode

and objects.

Techniques for plots interweaving

This section reviews some feasible techniques for plot interweaving taken from the Literary and Narratological studies.

The *"Chinese Box"* technique (Menéndez 2013) consists in the inclusion of nested stories inside a larger one. Every nested story can be related to one or several characters of the main plot line, or even consist of a separate one with the purpose of explaining some happenings of the main story. This approach has been applied in great works of literature such as *The Arabian Nights* (Vernet 1990) and *Don Quixote* (Cervantes 2011).

The technique of the *"Communicating Vessels"* (Menéndez 2013) is based on constructing a story by alternating at least two differentiated parallel plot lines. An example of this technique can be seen in Madame Bovary (Flaubert 1857), which contains a chapter that alternates two apparently disconnected plot lines. The only commonality between them is the temporal context in which the actions and happenings they contain are occurring. The resulting effect is a contamination between the two plot lines that, taken in isolation, would produce a different understanding in the reader. In other words, when two alternating story passages are interwoven together, proximity and alternation generate mutual influence. This influence can be applied to the tone, the tension, or the atmosphere that every story transmits to the other.

However, this technique is not limited to these types of influence. A stronger influence is also possible at the plot level. That means that both lines can develop a closer connection and converge at a certain point in the story. This can be achieved by using shared characters in both plots, acting as a hook between them.

Regardless of how strongly connected the plot lines are, it is important that there is a balance between the two plots, in order to avoid that one predominate over the other.

Despite the prior explanation has focused on the application of the technique on only two plots, it is also applicable to more than two plot lines. An example of several overlapping plots in which the characters intersect is Pulp Fiction (Tarantino 1994).

Proposed solution

This section focuses on detailing the approach adopted for implementing the multiple plot interweaving techniques described above and the consequent evolution of the existing architecture of INES, putting special stress on the new central component of the solution: the Plot Weaver service.

Multiple-plot generation process

The structure of a story plot in INES is based on a sequence of scenes, each of which is defined by a set of preconditions, a set of postconditions, the time in which it happens and a spatial reference in the story space (Concepción, Gervás, and Méndez 2018a; 2018b). According to this model, the weaving of scenes from different plots should take into consideration the logical consistency when combining their respective pre and postconditions. Following this reasoning, there are two ways of weaving scenes from different plots: combination and juxtaposition. The first strategy combines two scenes from different plots into a new one. This operation is feasible if and only if both the preconditions and the postconditions from the two scenes are respectively consistent among themselves from a logical point of view. The juxtaposition approach creates a combined sequence of scenes by putting one after another. In this case, the postcondition of every scene must be consistent with the precondition of the scene that goes after it. The latter is the approach adopted for the design of the Plot Weaver. This microservice, that will be concisely described later, is responsible for implementing the interweaving of the plot lines.

So, in order to weave the plots, the preconditions and the postconditions of the involved episodes must be consistent. If not, the Plot Weaver skips one episode in the plot line and tries to match with the next one. There is always a chance that the plot lines are simply incompatible so the response in this case would be an error, and the Story Director would have to select a different pair of plots to merge.

The new generation process starts with the Story Director requesting the Plot Generator to create a first plot. The request includes as parameters a template, the characters' list and a theme. The *template* parameter is the logical name which references the plot template that must be instantiated to create the plot. This instantiation is strongly linked with the *theme* parameter, that represents the setting in which the story will take place. It is also a logical name referencing a particular context in the knowledge base. For example, when applying a plot template such as "The Destructive Outsider" summarised in Table 1, the "Community" can be a "Nineteenth Century Western North American Town" or a "Middle-Class Family", depending on whether the theme is "Far West" or "Family Drama".

The generated plot is the skeleton of a story draft that is persisted in the Draft Repository. This draft contains information about the setting —time and space in the story world–, a list of the characters mapped with the roles required by the template and the theme, and of course the plot line.

Following this first plot generation, the Story Director requests the Plot Generator to create a second plot. This time, the request includes as parameters a new characters' list containing a subset of the characters' list of the first plot, their roles, the setting —time and space–, and the theme of the existing draft. In addition, the Plot Generator will need to

locate a proper template which can fit the character roles required, according to the theme. In order to do do this, it must query the template repository to get all the available templates that consider the involved roles. If no match is found, then the characters' list will not contain any common character with the prior plot line. This case will produce a story with two parallel plot lines that take place in the same setting, but apparently unrelated.

The procedure described above shows how the Story Director keeps the global consistency between the two generated stories. After having generated the two plots, the weaving process can start. The Story Director requests the Plot Weaver to merge the plots of the two generated stories.

The weaving of the plots is performed according to a sequence of steps. Taking the plot of the first story as the master line, the Plot Weaver proceeds by finding a compatible scene in the second plot line and merging it with the first one to insert a new scene in the master plot line. This means that the weaving process always takes the sequence of episodes of this first plot and proceeds by looking for compatible episodes in the second plot. The ideal outcome of this procedure is the generation of an interwoven plot line consisting in an alternation of scenes from the first and the second plot. The limitation of this strategy is that non-merged scenes from the source plots could remain, which would be included at the end of the resulting plot.

Evolved INES architecture

Adding the objective of supporting multiple-plots generation to the current operational requirements, the original generation model is insufficient. This need adds a new functional driver to the existing architecture. The adaptation of INES for interweaving plot lines is based on the definition of a set of plot merging heuristics and the modification of the INES microservices ecosystem to include this stage in the generation process. Figure 2 depicts the current architecture after evolving INES to support multiple plot generation.

The original INES model considered only one plot per story. This entails that the Story Director only requests the Plot Generator for a plot line once for every story. In the evolved version, the Story Director requests twice the Plot Generator to get the plot lines to merge and must include new controls along its inner logic to prevent inconsistencies.

As described previously, the original REST interface of the Plot Generator supported requests with no parameters, so it instantiated a random template, and requests with the template to be applied to generate the plot. In this case, generating a consistent story requires that the different plots share a common story space. This entails not only location and time, but also the set of participating characters. So that, the REST interface of Audrey —the Plot Generator, has been modified to accept all these new parameters. In order to adapt the architecture for mixing two plot lines into a single story, it is also necessary to include a new component. However, the adaptation of this microservice has entailed more changes than merely adapting its interface. Audrey uses a inner template repository for managing the templates it instantiates during the plot generation. The original design of this component only allowed for querying templates by

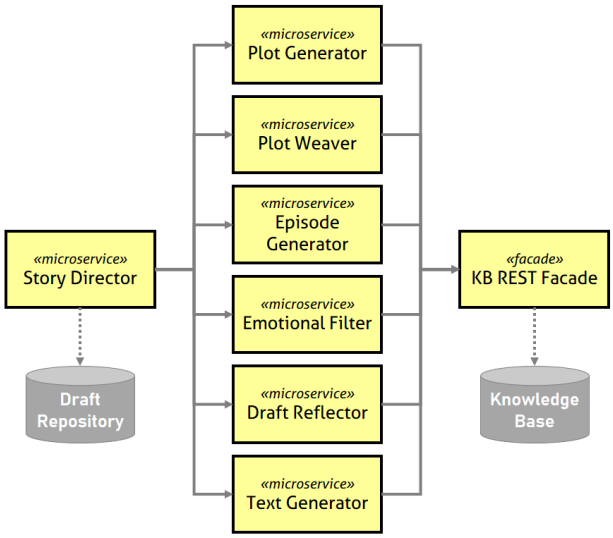


Figure 2: The evolved architecture of INES.

name. Due to the need for having a way of selecting templates by the roles they involve, the signature of the template repository has been adapted to provide queries based on roles. Thus, the updated Plot Generator can choose a template to apply according to certain restrictions. This functionality is essential to implement the multiple plot generation procedure.

The Plot Weaver is the microservice devoted to perform the plot weaving stage. It is implemented according to the Strategy pattern in order to select and apply the selected weaving heuristic. Initially, there have been considered two heuristics derived from the "Comunicating Vessels" technique. The simplest way of mixing the plots is by a mere alternation of episodes, a kind of "unrelated juxtaposition" of episodes from the two plot lines. In this basic case, the sets of characters of the two original plots can be disjointed. A more elaborated way is the linking of the episodes according to their compatibility in terms of state of the story world. In this case, there are common characters among the two stories.

The Plot Weaver provide as the default strategy the alternation of episodes from the two plots to combine –by juxtaposition. The resulting plot line will be a sequence of episodes picked from the two initial plots. Beside this, it also provides the option of merging the plots by sharing a subset of every plot's characters. This operation is the most complex and requires the Story Director to share certain parameters during the plot generation stage. So, it has to request the Plot Generator twice, for generating the two plots to combine, and provide shared information as parameters:

- Setting reference, as mentioned before, in order to instantiate the plot template according to a particular setting in the knowledge base, the Plot Generator requires a reference for this setting —e.g. "Far West", "Present day", "Epic Fantasy".

Episode	Description
Initial state	A frustrated character regrets his / her fruitless life
Temptation	The character is tempted by another character representing the evil forces
Pact with evil	The main character agree to serve the evil cause in exchange for a new satisfactory way of life
Evil actions	The main character performs evil actions induced by evil
Enlightenment	The main character becomes aware of being enslaved by evil
Redemption	The main character performs a saving action and dies. Evil is defeated

Table 3: The "Faust" story plot template

- Characters list, containing their name and their role in the plot template. The Story Director has to select which characters will be shared between the two plots in order to request the Plot Generator to include them.

In addition, the Story Director must take care of not requesting the Plot Generator using the same theme —e.g. requesting to generate two plots for interweaving based on "The destructive outsider", and choosing a compatible plot template for the second plot.

The interface of the Plot Weaver is also a stateless REST-based API, as the rest of the microservices of INES. In a first version, the Plot Weaver is only capable of combining two plot lines. An obvious precondition for these two plots to be merged is that they share a subset of the characters involved in their respective plot lines. For assuring this precondition, the Story Director must analyse the characters' roles of the first plot before requesting the Plot Generator for the second time. It needs to identify a set of matching roles between the first plot line and any of the available plot templates for the second plot line. For this reason, the REST interface of the Plot Generator includes a new operation for requesting information about the available plot templates and their metadata —such as characters' roles.

An example of interwoven plot story

The following lines introduce an example of story generation by plot interweaving according to the "Communicating Vessels" technique. It is structured around a first plot based on the "Destructive Outsider" template and a second one based on the "Faust" template. Tables 1 and 3 show the detail of both plot templates.

Table 4 shows a sample story which combines the two plots, generated by applying a juxtaposition approach with shared characters. The story combines two plots based on two different templates. The white rows contains the episodes from the first plot —based on "The destructive outsider" template, while the gray rows contains the episodes from the plot based on "Faust". Both plots share the same setting and

Episode	Actions
A peaceful community	Mary and John work together on their farm William helps John with the farm tasks Mary invites William to dinner Jeff visits John Jeff gives a present to John
Frustrated character regrets his life	Jeff feels miserable Jeff thinks that he is weak Jeff hates Carlson Jeff wants to arrest Carlson
The arrival of the outsider	Adam arrives at the city Adam buys a ranch Jeff welcomes Adam John welcomes Adam Mary invites Adam to dinner
Temptation	Adam offers help to Jeff Adam offers money to Jeff Adam tells Jeff to arrest all the gunmen
Outsider destructive actions	Adam wants John's farm Adam sneakily burns down John's barn
Pact with evil	Adam blames Carlson for burning down John's barn Jeff accepts Adams' money Jeff arrests Carlson
Conflict	Adam offers Mary to buy her farm Mary accepts Adam's offer John refuses to sell his farm John gets angry with Mary
Evil actions	Adam shoots John John is injured
The outsider revealed	Mary witnesses Adam shooting John Mary tells Jeff that Adam is a killer John tells Jeff that Adam is a killer Jeff gets angry with Adam
Enlightenment	Jeff realises that Carlson did not burn out John's barn Jeff releases Carlson Jeff says sorry to Carlson
The rise of the heroes	John faces Adam John demands Adam to leave Adam refuses to leave the town
Redemption	Jeff arrests Adam Adam shoots Jeff Carlson shoots Adam Adam dies Jeff dies
Conclusion	Mary says sorry to John Carlson is freed

Table 4: A sample story based on mixing two plots

a subset of the characters. The story takes place in the Far West and the characters of the whole story are the following:

- John, a farm owner married with Mary
- Mary, John's wife
- William, a farmer friend of Mary and John
- Jeff, the Sheriff
- Carlson, a gunman
- Adam, a cattle baron, an outsider

William is a character that only appears in the first plot, while Carlson only appears in the second one. Despite this, the combined plot line remains consistent.

This example shows the most complete form of combination that the current design can support. The two plot lines involved in the generation of the story contained a good number of shared compatible roles, so the outcome looks quite united. In addition, the scenes from the first plot perfectly alternate with the scenes from the second one. In this case, the postcondition of every scene from the first plot has been compatible with the precondition of the equivalent from the second plot, but this is not necessarily the norm. In many cases, the Plot Weaver will need to pass to the next scene in the first plot until it can insert the scene in the second. Moreover, the number of scenes in the plots to interweave can perfectly be different, and this will entail that the mixed plot line will contain several consecutive scenes from the same plot line.

It is also worth to mention that, in many cases, the preconditions and postconditions basically refer to facts that rarely affect to scenes from different plot lines. This means that there could be many combinations in terms of scene ordering that also would keep the global consistency of the story. The resulting ordering is mainly due to the previously described interweaving procedure, which always takes first a scene from the first plot and tries to find the next compatible scene in the second plot. Table 4 shows several examples of this. For example, the scenes "A peaceful community" and "Frustrated character" are interchangeable without affecting the consistency of the story, as well as the "Outsider destructive actions" and "Pact with evil". On the other side, the "Temptation" scene will be pointless if it happened before the "Arrival of the outsider". In this case, the Plot Weaver would have applied the procedure to establish a consistent ordering of the scenes.

Conclusions and future work

The presented adaptation entails a good number of validations in terms of knowledge representation and consistency. The assurance of a consistent merging of two different plot lines, putting together episode by episode from the two plots, is not an easy task. Despite the Plot Weaver checks the proper fitting of the respective precondition and postcondition of the episodes, there can be inconsistencies at a global level.

A significant case that can easily occur in the current model is the reappearance of a character killed in an episode of one plot in later episodes of the other plot, creating a kind of blocking inconsistency —the affected plot could not continue in a consistent way with the merged plot. This is caused by the fact that, despite the match of the corresponding preconditions and postconditions, the current version of the Plot Weaver does not consider the whole plot line, so there can emerge inconsistencies from a global point of view. For example, in one of the two plot lines, a common character can die. It is perfectly possible that this character does not appear in the episode that follows the one in which he / she dies. This circumstance makes the postcondition of the first episode to be consistent with the precondition of the following one. But, the character suddenly appears later, in an episode from the plot line which did not include the death of this character. This situation can be amended by including long-term conditions, that are propagated across the whole plot lines. In further iterations, these checks will be faced to guarantee a fully consistent story. On the other hand, from a positive point of view, this kind of situations could be interesting for developing stories according to an unnatural narrative plan, what could be specifically explored in a future line of research. The next natural step in this adaptation process will be the development of the mechanism that holds this need. The evolution of this model will provide generated lessons that will be helpful for making better decisions, and in marking paths for future investigation. One of this paths can be deepening in the development of a more pervasive plot interweaving, in which a Plot Weaver works with incomplete drafts. In this scenario, the plot generation and the episode generation stages in the different generation stages would directly interchange events between them, during their own activity. This approach, more decentralized in the sense of the events will not be mediated by the Story Director, would bring more creative wealth, but also more complexity to the process.

The Plot Weaver supports the application of different weaving strategies. One of the promising candidates to be considered in future versions is the "Chinese Box" technique (Menéndez 2013), which considers the development of nested plot lines inside a larger one. This adaptation would entail not only a modification of the Plot Weaver strategy, but also the model of generation applied by the Plot Generator.

Another aspect that can be analyzed in the future is the ability of merging more than two plot lines in a single story. Many of the already designed heuristics will still be useful, but probably we would need to design new ones to address the complexities associated to this new requirement and introduce more thorough draft evaluation mechanisms (Gervás and León 2016; Tapscott et al. 2016).

Acknowledgments

This paper has been partially funded by the projects IDiLyCo: Digital Inclusion, Language and Communication, Grant. No. TIN2015-66655-R (MINECO/FEDER) and InVITAR-IA: Infraestructuras para la Visibilización, Integración y Transferencia de Aplicaciones y Resultados de Inteligencia Artificial, UCM Grant. No. FEI-EU-17-23.

References

Alber, J., and Heinze, R. 2011. *Unnatural narratives-unnatural narratology*, volume 9. Walter de Gruyter.

Balló, J., and Pérez, X. 2007. *La semilla inmortal: los argumentos universales en el cine*. Ed. Anagrama.

Bellifemine, F.; Poggi, A.; and Rimassa, G. 1999. Jade–a fipa-compliant agent framework. In *Proceedings of PAAM*, volume 99, 33. London.

Bringsjord, S., and Ferrucci, D. 1999. *Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine*. Psychology Press.

Cervantes, M. 2011. *Don Quixote*. Penguin Random House.

Cohen, P. R., and Feigenbaum, E. A. 2014. *The handbook of artificial intelligence*, volume 3. Butterworth-Heinemann.

Concepción, E.; Gervás, P.; and Méndez, G. 2017. A common model for representing stories in automatic storytelling. In *6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017*.

Concepción, E.; Gervás, P.; and Méndez, G. 2018a. Afanasyev: A collaborative architectural model for automatic story generation. In *5th AISB Symposium on Computational Creativity. AISB 2018*.

Concepción, E.; Gervás, P.; and Méndez, G. 2018b. Ines: A reconstruction of the charade storytelling system using the afanasyev framework. In *Ninth International Conference on Computational Creativity, ICCC 2018*.

Dehn, N. 1981. Story generation after tale-spin. In *IJCAI*, volume 81, 16–18.

Fay, M. P. 2014. *Driving story generation with learnable character models*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Flaubert, G. 1857. *Madame Bovary*. Michel Lévy.

Gervás, P., and León, C. 2014. The need for multi-aspectual representation of narratives in modelling their creative process. In *5th Workshop on Computational Models of Narrative*, OASIcs-OpenAccess Series in Informatics.

Gervás, P., and León, C. 2016. Integrating purpose and revision into a computational model of literary generation. In *Creativity and Universality in Language*. Springer. 105–121.

Gervás, P. 2012. Story generator algorithms. In *The Living Handbook of Narratology*. Hamburg University Press.

Gervás, P. 2018. Storifying observed events: Could i dress this up as a story? In *5th AISB Symposium on Computational Creativity*. University of Liverpool, UK: AISB.

Glassner, A. 2009. *Interactive storytelling: Techniques for 21st century fiction*. AK Peters/CRC Press.

Lebowitz, M. 1984. Creating characters in a story-telling universe. *Poetics* 13(3):171–194.

Li, B., and Riedl, M. O. 2010. An offline planning approach to game plotline adaptation. In *AIIDE*.

Liu, H., and Singh, P. 2002. Makebelieve: Using commonsense knowledge to generate stories. In Dechter, R., and Sutton, R. S., eds., *AAAI/IAAI*, 957–958. AAAI Press / The MIT Press.

Machado, I.; Paiva, A.; and Brna, P. 2001. Real characters in virtual stories. In *International Conference on Virtual Storytelling*, 127–134. Springer.

Martínez, A. 2011. *Escribir teatro*. Barcelona, Spain: Alba Editorial.

Meehan, J. R. 1977. Tale-spin, an interactive program that writes stories. In *In Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, 91–98.

Méndez, G.; Gervás, P.; and León, C. 2014. A model of character affinity for agent-based story generation. In *9th International Conference on Knowledge, Information and Creativity Support Systems, Limassol, Cyprus*, volume 11, 2014.

Méndez, G.; Gervás, P.; and León, C. 2016. On the use of character affinities for story plot generation. In *Knowledge, Information and Creativity Support Systems*. Springer. 211–225.

Menéndez, R. 2013. *Cinco golpes de genio*. Barcelona, Spain: Alba Editorial.

Mott, B.; Lee, S.; and Lester, J. 2006. Probabilistic goal recognition in interactive narrative environments. In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, 187.

Pemberton, L. 1989. A modular approach to story generation. In *Proceedings of the fourth conference on European chapter of the Association for Computational Linguistics*, 217–224. Association for Computational Linguistics.

Perez y Perez, R. 1999. *MEXICA: A Computer Model of Creativity in Writing*. Ph.D. Dissertation, The University of Sussex.

Porteous, J.; Charles, F.; and Cavazza, M. 2016. Plan-based narrative generation with coordinated subplots. In *European Conference on Artificial Intelligence (ECAI 2016)*, volume 285, 846–854. IOS Press.

Prada, R.; Machado, I.; and Paiva, A. 2000. Teatrix: Virtual environment for story creation. In *International Conference on Intelligent Tutoring Systems*, 464–473. Springer.

Propp, V. 1968. Morphology of the folk tale. 1928.

Riedl, M. O., and Young, R. M. 2010. Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research* 39(1):217–268.

Tapscott, A.; Gomez, J.; León, C.; Smailovic, J.; Znidarsic, M.; and Gervás, P. 2016. Empirical evidence of the limits of automatic assessment of fictional ideation. In *C3GI@ ESSLLI*.

Tarantino, Q. 1994. Pulp fiction.

Todorov, T. 1975. *The fantastic: A structural approach to a literary genre*. Cornell University Press.

Vernet, J., ed. 1990. *The Arabian Nights*. Editorial Planeta.

Winston, P. H. 2016. The genesis story understanding and story telling system a 21st century step toward artificial intelligence. Technical report, Center for Brains, Minds and Machines (CBMM).

5.4. Artículo: Exploring Baselines for Combining Full Plots into Multiple-plot Stories

Este artículo es una versión más extensa del funcionamiento de todos los mecanismos de generación multitrama implementados en la versión evolucionada de INES. Introduce además una comparativa cualitativa de las dos técnicas propuestas, los *vasos comunicantes* y las *cajas chinas*, frente a una línea base formada por las estrategias de entrelazado aleatorio y de yuxtaposición alternante. El artículo recoge además una buena colección de historias generadas por aplicación de las diversas técnicas, con un número variable de tramas.

5.4.1. Cita completa

E. Concepción, Gervás, P., and Méndez, G. (2020) *Exploring Baselines for Combining Full Plots into Multiple-plot Stories*, New Generation Computing, vol. 38, pp. 593-633.

5.4.2. Resumen original

Many of the stories at the core of narrative entertainment involve a number of plot lines that combine to give them interest. The present paper sets out to solve the problem of how several different plot lines, each one of them complete in its own sense, can be combined into a single linear sequence that works reasonably well as a plot. Starting from a brief review of how existing storytelling systems address the task, a representation for plots and plot templates is proposed that allows the combination of several subplots into a single plot line. Four strategies for weaving plots are proposed, two taken from literary studies and two computational baselines, and a formative evaluation of a set of stories produced by these solutions is presented. Finally, open issues, promising avenues of future work and the relation to previous work are discussed.

New Generation Computing (2020) 38:593–633
https://doi.org/10.1007/s00354-020-00115-x

NEW GENERATION COMPUTING

Exploring Baselines for Combining Full Plots into Multiple-plot Stories

Eugenio Concepción[1] · **Pablo Gervás**[2] · **Gonzalo Méndez**[2]

Received: 18 March 2020 / Accepted: 9 October 2020 / Published online: 28 October 2020

Abstract

Many of the stories at the core of narrative entertainment involve a number of plot lines that combine to give them interest. The present paper sets out to solve the problem of how several different plot lines, each one of them complete in its own sense, can be combined into a single linear sequence that works reasonably well as a plot. Starting from a brief review of how existing storytelling systems address the task, a representation for plots and plot templates is proposed that allows the combination of several subplots into a single-plot line. Four strategies for weaving plots are proposed, two taken from literary studies and two computational baselines, and a formative evaluation of a set of stories produced by these solutions is presented. Finally, open issues, promising avenues of future work and the relation to previous work, are discussed.

Keywords Artificial intelligence · Computational creativity · Automatic story generation

Introduction

Many of the narrative forms that feature highly for their entertainment value in Western culture (Shakespearean comedies, Hollywood movies, TV series, and nineteenth century novels) share the characteristic of involving a number of plot lines that combine to give them interest. Many of these established forms rely on mechanisms for

✉ Eugenio Concepción
econcepc@ucm.es

Pablo Gervás
pgervas@ucm.es

Gonzalo Méndez
gmendez@ucm.es

[1] Facultad de Informática, Universidad Complutense de Madrid, Madrid, Spain

[2] Instituto de Tecnología del Conocimiento, Universidad Complutense de Madrid, Madrid, Spain

artfully combining several narrative threads into a single linear sequence, so that audiences can easily follow the overall story.

Given the relative importance of this mechanism for so much of our ability to communicate, it is surprising to find how little research has focused on understanding the computational principles that govern it.

The present paper sets out to solve the problem of how several different plot lines, each one of them completes in its own sense, can be combined into a single linear sequence that works reasonably well as a plot. An example of the type of combination desired is the popular solution for a Hollywood plot that combines an impending catastrophe that threatens to destroy the world, the attempts of several characters to survive the coming cataclysm, insights into the personal love interests of some of these characters, and often the recovery of damaged family relations between them.[1]

The paper relies on an existing storytelling system capable of generating single plots [8] to construct stories by combining more than one plot line. For the purpose of exploring the possible ways of creating stories in such way, an analysis on several plot combination strategies has been performed, considering four techniques that support the weaving of previously generated plots to create a single story line.

The paper begins with a revision of the literary historical background and some previous works on this subject in automatic story generation. Section 2 describes how existing storytelling systems address the issue under consideration. Section 3 provides a brief explanation of the story representation aspects relevant to understand the proposed solution. Section 4 describes the operation of the story generation system and the algorithmic aspects of the techniques applied to weave subplots together, and presents some examples of stories generated by means of these algorithms. Section 5 outlines a qualitative evaluation of a small set of generated stories. Section 6 discusses insights from the formative evaluation, open issues, outlines some promising avenues of future work, and analyzes the relation to previous work. Section 7 draws conclusions from the material presented and highlights the most relevant findings.

Related Work

This section focuses on reviewing related work that is relevant to the purpose of this research. It starts by presenting some relevant antecedents in automatic storytelling and then proceeds to review research efforts that have focused specifically on multiple-plot generation.

The relation of these antecedents to the work presented in the paper is discussed in Sect. 6.3, to allow reference to the material presented in the paper.

[1] Think *Independence Day* (1996), *Armaggeddon* (1998), *The Day After Tomorrow* (2004), or *2012* (2009) to name but a few.

Relevant Antecedents in Automatic Storytelling

The similarities between a story and a plan (both have an initial and a final state, and a set of causally linked steps that drive from one to the other) have led to the appearance of many efforts to address automatic storytelling by means of AI planning solutions [4]. Such solutions are based on the concept of a *planning operator*, a unit of representation that represents an action, which states explicitly a set of preconditions of the action to happen and a set of postconditions that hold when it has happened. Planners rely on these units to build chains of causally related action from an initial state to a goal. A pioneer in this line was TALE-SPIN [22], one of the first story generators, that wrote short stories about the inhabitants of a forest. Although the concept of plot was not explicitly mentioned in the reports on TALE-SPIN, the nature of its underlying representation of stories as plans results in an implicit structure to the stories, all generally based on the pursuit of a single specific goal by the protagonist character. They, therefore, were restricted to single-plot stories. An important improvement in automatic storytelling came about with Author [11], a storytelling system also based on planning but which considered not only character goals but also an additional level of authorial goals. Whereas the goals of characters tend to draw a straight line through a story, from initial situation to successful outcome, the goals of authors tend to focus on inserting obstacles in the path of the protagonist—to make the story interesting to the reader. Sometimes, these obstacles take the form of rival characters with conflicting goals, which sometimes may lead to the insertion of additional plot lines. These initial systems attempted to generate closed stories, with a clear beginning and a clear end. A different development was introduced in Universe [20], a system that generated scripts for a TV soap opera by focusing on the interactions between characters. By construction, Universe operated on an open-ended story, adding episodes on each run but always leaving the story open to be continued in subsequent episodes. This required an approach to generation based on a large cast of characters and the systematic introduction of complications affecting some of these characters to trigger the events of the new episode. As a result, the outputs of Universe over a sequence of runs—a set of episodes—may convey the impression of a multi-plot story. However, the generation processes employed at each point did not explicitly address the management of multiple plots, and it had no specific mechanisms for handling the interleaving of multiple-plot lines. The explicit consideration of the intentions of characters, to ensure that the actions of the characters in the resulting stories be believable, was the goal of the Fabulist [28] system, which presented a whole architecture for automatic story generation and presentation. This system combines both the author interests and the intentionality of the characters. The fact that different characters have different intentions, and that the system manages all of them successfully implies that many of the ingredients for the handling of multi-plot stories may already have been in place in this system, but the specific aspects of plot interweaving were not addressed in the published reports of it.

The advent of grammars as representational tools to capture the structure of complex symbolic sequences triggered interest in the concept of story grammars [29, 33]. A number of automatic storytellers attempted to exploit this concept. **GESTER**,

GEnerating STories from Epic Rules, [24] was a rule-based story generation system that managed information about story structure in the form of a simplified version of a narrative grammar that captured the possible events and actors of the epic subgenre. A more elaborate example was Brutus [2], a system that generated short stories about betrayal. It used a very thorough knowledge model for representing the concept and implications of betrayal. It also provided a grammar-based generation model and a literary beautifier, which allowed it to generate high-quality stories, providing texts that could have been written by humans. In both of these systems, the story grammars involved constitute the formal representation of the structure of stories that such systems rely on. In that sense, these grammars inherently capture the structural complexity of the subset of the genre they represent. Where this subset contains examples of multi-plot stories, the grammar will capture the essence of this feature, and it will allow the generation of stories of this type. However, such representations of multi-plot occurrences in a grammar are likely to be instances of one possible interleaving of plots rather than procedures applicable in general to interweaving problems.

A third possible approach to automatic storytelling is to follow theoretical models of how humans address the task. This approach was followed by MEXICA [25], a storytelling system that generated mythological stories about the Mexicas, the early inhabitants of Mexico. MEXICA was based on the engagement and reflection model of the writing task [30]. It included a reading module capable of processing a corpus of previous stories from which it obtained the knowledge structures it relied upon to generate new stories. The operation of the system is complex, but essentially relied on building for each prior story an interpretation in terms of the evolution of emotions of and tensions between characters, and abstracting from these representations a set of units of plot-building blocks that encoded the likelihood of two events occurring one in the context of the other in a story. MEXICA did not have an explicit concept of plot or plot line. The nature of its operation allowed for chains of events involving different characters to be combined into a story, which resulted in the possibility of multiple-plot line stories of a simple type to appear among its outputs. However, the system as such did not address any of the challenges of explicitly handling and interweaving multiple plots.

Multiple-Plot Generation in Automatic Story Generation

There is not a long record of multiple-plot generation and combination in automatic story generation.

One of the remarkable works in this respect is the plot weaving algorithm proposed by Fay [12]. Fay's system relies on a corpus of stories written in simple English that it can read and parse into a set of character models and a set of narrative threads for the various characters. Based on this material, the system can be asked to generate stories combining characters of a given type. It operates by finding the character models best matching the given types, retrieving narrative threads associated with the corpus with those models, and finding the best combination of those narrative threads into a single story. Fay's system reads stories

using the *Genesis* story understanding system [36]. The narrative threads for the chosen characters are woven together using an algorithm that makes sure that characters' plots are compatible and takes care of building a consistent timeline for all of the plot elements of the story. The actual procedure is computationally difficult, because it entails selecting the best set of bindings for the main characters in the story request to generic entities appearing as secondary characters in threads for other characters. These bindings are then used to determine the order in which the elements from each thread are combined into a single linear story.

Porteous et al. [26] set out to address the challenge of constructing stories with multiple interleaved subplots—as featured regularly in serial dramas and soaps—in the context of an interactive storytelling system. Their system relies on a plan-based approach to multi-plot narrative generation which successfully generates narratives conforming to different input parameters that specify aspects such as the number of subplots to be interleaved and the relative time spent on presentation of each subplot. In contrast with Fay's solution, which operated over a set of already determined plots obtained from prior stories, this solution constructs the various subplots it considers incrementally at the time of interweaving them. In this approach, a subplot is considered to be a sequence of *segments* of various types centred around particular character: usually, an *introduction* followed by a variable number of pairs of *obstruction* and *resolution*, with instances of *exposition*—used to provide additional information to the audience at need—optionally interspersed between them. A multi-plot story is a sequence of such segments in which adjacent segments are from different subplots. This shows some affinity with Fay's definition of plot, in as much as it is also focalised on a particular character, but it includes a significantly higher number of structural restrictions at the narrative level. The system operates by starting from a set of goals for each subplot to consider, and then incrementally building and presenting segments for the subplots, switching from one to another under the guidance of the input parameters.

The Raconteur system [15] focuses on the slightly different problem of generating a story to match a given set of observed facts. The facts to consider come as a set of time-stamped predicates describing the movements of pieces in a chess game. The system builds a set of narrative threads, each grouping the predicates affecting a given piece (either because it is involved in the action or the action takes place within its range of perception—within squares of the board sufficiently close to it). Stories are then built by: (1) selecting a set of pieces, (2) selecting particular spans of their narrative threads, and (3) weaving these spans together into a single linear sequence. The weaving is done by taking into consideration relative chronological order, physical proximity, co-occurrence of pieces across threads, and the nature of the actions captured in the predicates.

The StoryFire system [17] introduces a refinement into the process of thread weaving by considering explicitly the concept of plot. A plot in this context is a sequence of plot elements, each describing an event relevant to the structure of the story, and a set of roles that characters play in the event. Over a setup similar to that of the Raconteur system, spans of narrative threads for particular pieces are selected by finding optimised alignments of parts of the thread with plots from a database.

Then, the interweaving of threads is informed by considerations taken from the plots aligned with them at potential points for switching threads.

The PlotAssembler system [18] addresses the exploration of a large search space of acceptable plots by combining small spans of narrative discourse that do not necessarily appear contiguously in the sequence of a story. An example of this would be the kidnapping of a character at the start of a story that is related to the release of that character achieved towards the end of the story. Using a small set of such units, the process allows the construction of many different stories of arbitrary complexity, always satisfying constraints of coherence and providing a sense of resolution. By the nature of these segments—as bridges that span from an element at one point of the story to others not close to them—the construction procedure required for this process involves decisions on how to weave these segments together. The PlotAssembler system addresses this problem via a corpus-based solution, whereby probabilities of elements of different segments occurring together in a story linked by the presence of particular characters are mined from a corpus of prior stories.

Story Representation Aspects Relevant to the Plot Weaving Procedure

The purpose of this section is to establish the necessary context about those aspects used in the proposed solution, to enable a better understanding of the way the different plot combination strategies have been implemented.

The present paper focuses on the construction of plot, and does not address the details concerning the development of scenes nor the description of scenes as text. With respect to the traditional pipeline architecture for natural language generation systems [27], the present paper addresses issues of content planning, and it does not address aspects of sentence planning or surface realization.

This implies that the representations and the outputs discussed here should not be appraised in terms of the linguistic quality of the textual descriptions. Such descriptions are only provided to give an idea of the type of concepts that are being considered. In considering the quality of any given story in terms of the efforts reported in this paper, one should focus on their structural features from a narrative point of view, which are the only aspects that are being taken into account by the decision processes proposed. For details on how the specific procedures for the remaining stages of the pipeline operate for the underlying storytelling system, interested readers are referred to [7–9].

Key Definitions

This section provides a set of definitions of the key common concepts that will be referenced throughout the paper.

A story is the content of a narration. This statement surely entails a number of progressively more detailed stages ending up in a text that tells the story, but for the purposes of this paper, the relevant information about a story is represented

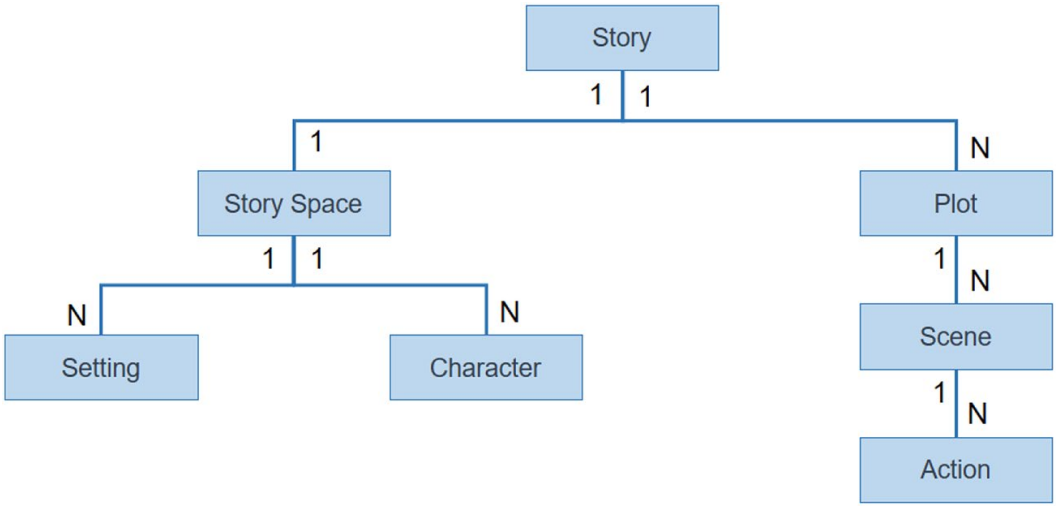


Fig. 1 The conceptual representation of a story

by means of a structure which includes two key elements: what happens—the plot and the setting in which the narrated action occurs—the story space [5]. The representation of a story is a hierarchical structure containing different levels of detail such as the plot as a sequence of scenes and the scene as a sequence of actions, as shown in Fig. 1. In addition to this visual summary, a complete example of a story is presented in Table 1.

Although the underlying storytelling system includes details on all of the concepts shown in Fig. 1, only the subset of those concepts that is relevant to the material in the present paper is described here. Additionally, for those concepts considered, only the details relevant to the combination of plots are described.

The characters represent the human persons, or other beings in certain cases, that interact in the story. The representation of a character in a complete storytelling system must consider physical, psychological, social, and cognitive-related characteristics, but the only feature that is relevant to plot combination at the level carried out in the paper is the role that each character plays in the story, expressed in terms of specific roles it plays in the subplots under consideration.

The plot is the basic structure of any story. For the purposes of the present paper, a plot is considered to be a collection of scenes arranged according to a structure which represents both the temporal ordering, and the causal relations between the actions performed and the things that happen in these scenes [5]. Once multi-plot stories are considered, the plot of a story will be a combination of subplots, interweaving the scenes from these subplots into a single linear sequence of scenes.

A scene is a unit of action that takes place in a particular location during a period of time. For the purposes of the present paper, a scene is composed by a sequence of actions performed by the characters in a specific location [5].

Every scene is accompanied by a combination of pre- and postconditions, which represent, respectively, changes in the global state of the story space before and after the scene happens. These conditions are expressed as a set of assertions related to the diverse characters and objects in the space.

Table 1 A sample single-plot story showing its main elements

Story	
Title	Beware of the fiancee
Story space (setting + characters)	
Setting	North America mid-20th Century Family
Character	Role
Anne, the new fiancee of Scott	Outsider
Edward, a wealthy businessman and head of the family	Community Member
Sarah, the elder daughter of Edward	Community Member and Hero
Scott, the youngest son of Edward	Community Member
Samuel, a friend of Scott	Community Member
Plot (based on template "The Destructive Outsider)"	
Scene	Actions
Scene 1: A peaceful community	Scott and Samuel work together on Edward's factory. Edward is proud of Scott. Sarah secretly loves Samuel. Edward, Sarah and Scott live together in a villa. Samuel visits Scott
Scene 2: The arrival of the outsider	Scott introduces Anne to Sarah and Edward. Scott gives a ring to Anne. Edward welcomes Anne. Sarah welcomes Anne. Edward invites Anne to dinner
Scene 3: Outsider destructive actions	Anne steals a jewel from Edward's room. Anne tells Edward that Samuel stole a jewel
Scene 4: Conflict	Edward accuses Samuel of stealing the jewel Sarah defends Samuel. Samuel gets angry with Scott
Scene 5: The outsider revealed	Sarah witnesses Anne trying to steal a swatch. Sarah tells Edward that Anne is a thief. Edward tells Scott that Anne is a thief. Scott gets angry with Edward
Scene 6: The rise of the heroes	Sarah faces Anne. Scott demands Anne to leave. Anne leaves the town
Scene 7: Conclusion	Scott says sorry to Edward. Edward says sorry to Samuel. Samuel gives thanks to Sarah. Samuel falls in love with Sarah. Samuel ask Sarah to marry him

To illustrate how these pre- and postconditions have been considered for the purpose of this research, a sample scene is shown in Table 2 (corresponding to scene 3 from the story presented in Table 1). In this scene, the character Anne steals a jewel and blames Samuel. The preconditions of the scene allow for such events to happen and the postconditions reflect their results.

A plot template is an abstract description of an internally coherent unit of plot material that represents a potential subplot and which can be used as a building block for more elaborate plots. It basically consists of an arrangement of generic scenes in which a list of characters participate in certain roles—relevant to that

Table 2 Example of a scene with preconditions and postconditions indicated

Preconditions	Edward is a friend of Anne ; Edward trusts Anne ; Edward has a jewel in his villa ; Anne is in Edward's villa ; Anne is evil ;
Actions	*Outsider destructive actions*; Anne steals a jewel from Edward's room ; Anne tells Edward that Samuel stole a jewel ;
Postconditions	Anne has Edward's jewel ; Edward has no longer his jewel ; Edward is a friend of Anne ; Edward trusts Anne ; Anne is in Edward's villa ; Edward believes that Samuel is evil ; Edward is an enemy of Samuel ; Edward is upset with Samuel ; Nobody knows that Anne stole the jewel ; Anne is evil ;

Table 3 "The Destructive Outsider" story plot template

Template "The Destructive Outsider"	
Role	Description
Outsider	This character is a newcomer to the community. Initially, she/he performs friendly actions, but later her/his evil nature is revealed
Community member	Any of the characters playing this role behaves friendly with the others until the beginning of the outsider's evil actions, at which point the lack of trust becomes widespread
Heroe	Any of the characters playing this role is also a community member. Their behaviour differs from the rest of community members when they raise to face and defeat the evil outsider
Plot Structure	
Scene	Description
Peaceful community	Initial state introducing the members of a peaceful community
Arrival	The arrival of the outsider to the community
Outsider destructive actions	The outsider acts against the members of the community, performing destructive actions, without being uncovered
The outsider revealed	The true evil nature of the outsider is revealed
The rise of the heroes	The heroes rise from the community and fight against the outsider
Purge	The outsider is purged. The community becomes peaceful again

particular subplot—by performing certain types of actions. The roles and the types of actions they must perform constitute a set of constraints that the generated plot must meet. Table 3 shows an example of plot template that has led to generate a story such as the one summarised in Table 1.

Although it has obvious similarities to a plot, a plot template is conceptually different from a plot. A plot template specifies information that needs to hold at different points in a story for the plot template to be identified as being relevant to the structure of the story. The information in a plot template establishes a set of generic

scenes that need to be present, and a relative order between them, but it is generally agnostic as to what else may happen in the story in between the scenes captured in the plot template. This is what allows different plot templates to be woven together into stories. Once the structure of the plot for the story is fixed in terms of which plot templates take part in it and how they are woven together, the scenes provided by these plot templates for a given story—which correspond to subplots for the story—may also need to be "fleshed" out with details designed to give cohesion to the story as a whole. This process is important to the quality of the final story, but it is not addressed in the present paper due to constraints on space. However, once the structure of the story has been established by a finished plot, the process of fleshing out these scenes for a multi-plot story is no different from that for a single plot. Interested readers can find more details on this process in [8].

The concept of role is strongly linked to the characters and the plot generation process. For the purposes of the present paper, a role is defined as the function played by one or more characters in a particular subplot of a story. This function is necessary to determine the type of actions that this character should or can perform throughout the story.

The importance of the role is essential during the Plot Generation stage, because the fleshing out of a template takes as input a list of roles and information about the type of actions that the characters must perform at each scene. The roles condition the behaviour of the characters. There are roles that are optional. This means that it is possible that no character plays it in the generated plot. Typically, an optional role will relate to a secondary character in the story. Table 1 shows an example of a story characters and their corresponding roles. A detailed description of the roles participating in this story can be found in Table 3.

Constructing Stories by Combining Plots

The process of story generation considered in this paper involves the tasks of plot generation and plot weaving. These two tasks together constitute the content planning stage of the story construction process. To achieve a full story as a text in natural language, additional stages would be required that are not relevant for the goals of the present paper (Scene Generation, which corresponds to low-level detailed determination of scene content and detailed action planning, and NLG, which corresponds to Natural Language Generation). Readers interested in these additional stages of the finer details of knowledge representation as featured in the underlying storytelling system are referred to our prior work on the subject [5, 10].

Plot Generation Process

For the purposes of the present paper, plots are considered as inputs to be combined into multi-plot stories. Such inputs are constructed by a Plot Generator. The Plot Generator is a template-based engine which produces the plots from a subset of the cinematographic basic plots compiled by Balló [1], represented as plot templates. It uses a inner template repository for managing the plot template

catalogue. The examples presented in the paper have been built with a restricted set of plots that are shown in Appendix 9. The full system operates with a larger set of templates.

Once a specific template has been chosen, the Plot Generator generates a basic draft of a plot by instantiating the roles in the templates with characters. This instantiates the characters in the scenes in the plot, which is now ready to be used as a potential subplot in a further weaving stage. This process of character instantiation relies on knowledge about the context in which the story will be set. The detail of every scene will be completed later by the Scene Generator component. These additional processes are beyond the scope of the paper. For the purposes of the present paper, each such draft for a subplot can be considered to contain a list of the characters mapped with the roles required by the template and the sequence of scenes involved.

Plot Weaving

The construction of more complex stories can be achieved by combining two or more simple plots as subplots for a larger story. For simplicity, the examples in this paper consider only combinations of simple plots consisting of instantiations of a single-plot template, but in general terms, the plots to be combined can themselves be complex.

A Plot Weaver module receives as inputs a set of plots to be combined and the strategy to apply for combining them. The strategies available to the system in its present state are four (Random, Alternating Juxtaposition, Communicating Vessels, and Chinese Boxes) and they are described in detail in Sect. 4.

Two concepts need to be explained to understand the basics of plot weaving: the difference between weaving and merging scenes, and the mechanism for precondition checking.

Scene Weaving vs. Scene Merging

There are two possible ways of combining scenes from different plots.

One is to create a sequence of scenes where each scene of the sequence originally appeared as a scene in one of the constituent plots. Scenes from the different plots are related in the combined plot by reason of either their relative order in the sequence or any bindings established between the characters that appear in them that occur during the integration. This is what we call *scene weaving*, and it is the procedure followed in the present paper.

The other possible way of combining scenes is to consider the possibility of merging particular scenes from one plot with scenes from the other plot. This creates new scenes that were not in either of the original plots, but which share characteristics of scenes from both. This is what we call *scene merging*, and it is not addressed in the version discussed in the present paper.

Table 4 Potential match between scene S from plot A and scene B from plot B (pre- and postconditions shown in box in each case)

Scene S from Plot A	Scene T from Plot B
Mary plays with Nemo in the garden; Nemo accidentally kills Mary; Nemo discovers that he is not human; Nemo leaves home.	Benson is discovered by Mary while stealing money; Mary feels sad; Mary loves no longer Benson; Mary tells West that Benson is a thief; Mary hates Benson.
Mary is in the garden ; **Mary is dead** ; Nemo has killed Mary ; Nemo knows it is not human ; Nemo is not in the house ;	**Mary is in the house** ; **Mary loves Benson** ; Benson hates Mary ; Benson is in the house ;

Assertions that are inconsistent are marked in Bold

Table 5 Potential match between scenes S and S* from plot A (pre- and postconditions shown in box in each case)

Scene S from Plot A	Scene S* from Plot A
Mary plays with Nemo in the garden; Nemo accidentally kills Mary; Nemo discovers that he is not human; Nemo leaves home.	West looks for Nemo in the forest; West looks for Nemo in the town.
Mary is in the garden ; Mary is dead ; Nemo has killed Mary ; Nemo knows it is not human ; Nemo is not in the house ;	West knows that Nemo is not in the house ; Nemo is not in the house ; West wants to know where is Nemo ;

No inconsistencies identified

Precondition Satisfaction

Combining two plots involves deciding when a scene S1 from plot A can be placed in the discourse of a story directly after a scene S2 from plot B. The important criterion here is whether the appearance of the scenes in that order within a story is likely to give rise to inconsistencies. This is achieved by a checking of preconditions across scenes that are likely to end as neighbours in a story after a combination. This check of preconditions is ultimately based on a level of representation that is not covered by the material included in the paper, as it pertains to the tasks of enrichment of the details of the story that are associated with levels beyond content planning. For completeness, a brief description of the mechanics involved is given in this section.

The examples given in Tables 4 and 5 show the case where an inconsistency arises between the preconditions of scene T from plot B and scene S from plot A (Table 4) and the trivial case where there are no inconsistencies between the preconditions of scene S* from plot A and preceding scene S from plot A. The inconsistencies detected in the first case are:

– Mary is in the garden vs. Mary is in the house.
– Mary is dead vs. Mary loves Benson.

An Exploration of Strategies for Weaving Plot Lines

This section presents a catalogue of strategies for plot interweaving that were analyzed for this research. This is not supposed to be an exhaustive list of all possible approaches, but rather a concise analysis of those algorithms that could easily be adapted to a computational context and applied for the purpose of this research. The strategies proposed for weaving plots are applicable to sets of plots of any cardinality, starting from a minimum of two plots.

Weaving Strategies

Four strategies for weaving subplots have been considered in this paper and brief descriptions of each one of them are provided below (interested readers can find the pseudocode for each of the algorithms in Appendix 10):

– *Random Weaving*: The next plot from which to take the next scene is selected at random and the next scene available in it is added to the draft, with no checking of pre-/postconditions for consistency.
– *Alternating Juxtaposed Weaving*: The plots to combine are processed sequentially in a cycle. At each iteration, a scene is taken from each one of them and inserted into the draft with no checking of pre-/postconditions for consistency. Results in systematic ordered alternation between the scenes from different plots.
– *Communicating Vessel Weaving*: The plots to combine are processed sequentially in a cycle, at each point adding to the draft a scene from the next plot if it is compatible with the draft so far in terms of precondition satisfaction. Priority is given to inserting scenes from a different plot, but if no consistent option is found, the next scene from the same plot is added. Results in a weaving of consistent fragments of different sizes of the plots to be combined.
– *Chinese Boxes Weaving*: One plot is selected initially as main plot and added to the ongoing draft. The remaining plots to combine are processed sequentially. For each additional plot, the algorithm attempts to fit it between two scenes of the draft, based on whether the preconditions of the first scene and the postconditions of the last scene of the plot to be inserted are compatible with the postconditions of the previous scene and the preconditions of the next scene of the draft, respectively. Essentially, it checks if the gap between two scenes in the draft is suitable for inserting a whole secondary plot. This technique may lead to any number of levels of nesting of the subplots of a story, restricted only by the number of available subplots (and precondition satisfaction constraints).

The strategies for plot weaving are graphically summarised graphically in Fig. 2.

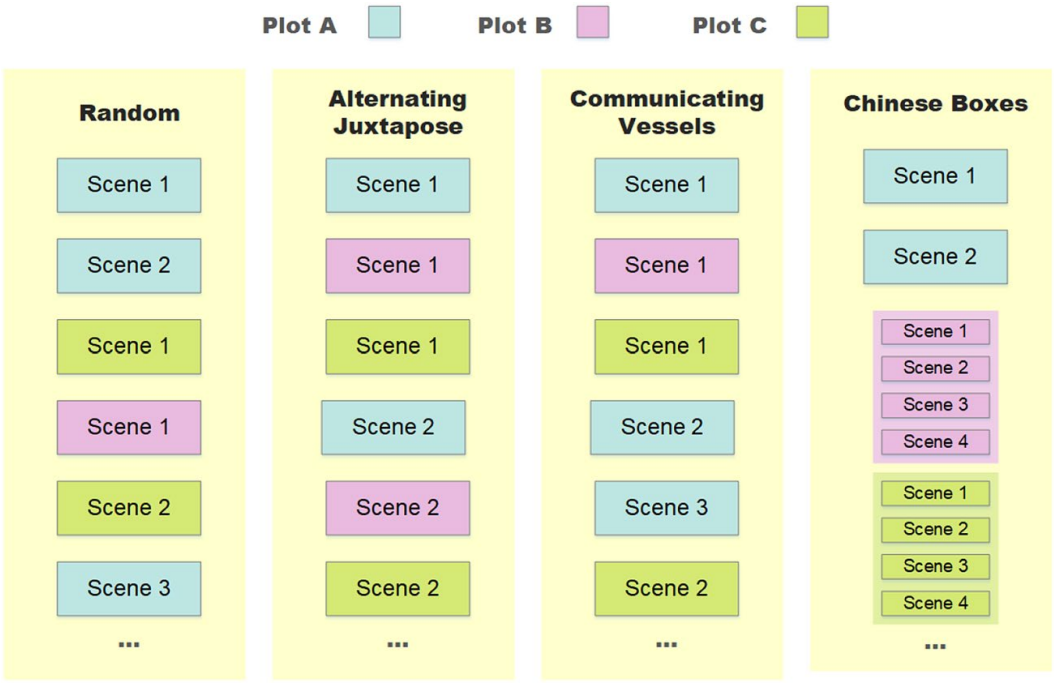


Fig. 2 Graphical summary of the operation of the different plot weaving strategies

The first two strategies are introduced as baselines for measuring the effectiveness of the other plot weaving strategies. The last two strategies are documented literary techniques which have been applied for centuries in the literature [23].

The technique of the *Communicating Vessels* [23] is based on constructing a story by alternating at least two differentiated parallel plot lines. An example of this technique can be seen in Madame Bovary [13], which contains a chapter that alternates two apparently disconnected plot lines. The only commonality between them is the temporal context in which the actions and happenings they contain are occurring. The resulting effect is a contamination between the two plot lines that, taken in isolation, would produce a different understanding in the reader. In other words, when two alternating story passages are interwoven together, proximity and alternation generate mutual influence. This influence can be applied to the tone, the tension, or the atmosphere that every story transmits to the other.

The Frame Story technique—or the *Chinese Boxes* technique as Menéndez names it [23]—relies on the inclusion of nested stories inside a larger one. Every nested story can be related to one or several characters of the main plot line, or even consist of a separate one with the purpose of explaining some happenings of the main story. This is a quite common literary technique that has been applied in great works of literature such as *The Arabian Nights* [34] and *Don Quixote* [3]. The point of the secondary story is to serve as a companion piece to another story, within this one, where an introductory or main narrative sets the stage for a more emphasized second narrative or for a set of shorter stories. This arrangement leads the reading from a first story to one or more stories within it. It can also serve to inform the readers about key aspects of the narrative that would otherwise be difficult to understand.

Examples of Interwoven Plot Stories

This section presents several examples of story generation by plot interweaving according to the discussed strategies.

Example with Two Plot Lines

Table 6 shows a sample story which combines two plots—one based on the "Destructive Outsider" template and one based on the "Faust" template—using the Communicating Vessels approach with shared characters. Details of the plot templates involved are given in Table 13 in appendix 9. The roles played by the characters in the various plots are shown in Table 7.

This example shows the most complete form of combination that the current design can support. The two plot lines involved in the generation of the story contained a good number of shared compatible roles, so the outcome looks quite united. Scenes from the first plot perfectly alternate with the scenes from the second one. This is not necessarily the norm, as consideration of constraints on precondition satisfaction applied by this technique may result in longer spans of each plot being inserted into the draft as continuous subsequences (see, for instance, the discussion of Story 2 in Sect. 5.4).

Example with Three Plot Lines

Table 8 shows a sample story which combines three plots based on three different templates. Details of the plot templates employed can be found in Table 13 in Appendix 9. There is only one shared character, Hawa. The roles played by the characters in the various plots are shown in Table 9.

Evaluation of Plot Weaving Strategies

The evaluation of story generators focused on content planning rather on generation of full-textual versions of the stories they construct presents well-known problems [14] due to a combination of features of these systems:

- they focus on generating an abstract representation of the structure of the story, in which it is often the relations between the elements rather than the actual elements being presented that is the goal
- the abstract representations are either difficult to read (if they focus on capturing the structure) or considerably removed from the information on which the system is focusing (if they are transcribed as readable text by ancillary processes distinct from the modules of the system performing the content planning)
- the outputs of these processes, by virtue of having focused on particular aspects to the detriment of others, tend to appear inordinately clumsy to the untrained

Table 6 A sample story generated by combining two plots based on "The Destructive Outsider" and "Faust" by applying the "Communicating Vessels" strategy

Destructive Outsider	Faust	Actions
Peaceful community		Mary and John work together on their farm. William helps John with the farm tasks. Mary invites William to dinner. Jeff visits John. Jeff gives a present to John
	Frustrated character regrets his life	Jeff feels miserable. Jeff thinks that he is weak. Jeff hates Carlson. Jeff wants to arrest Carlson
Arrival of the outsider		Adam arrives at the city. Adam buys a ranch. Jeff welcomes Adam. John welcomes Adam. Mary invites Adam to dinner
	Temptation	Adam offers help to Jeff. Adam offers money to Jeff. Adam tells Jeff to arrest all the gunmen
Outsider destructive actions		Adam wants John's farm. Adam sneakily burns down John's barn
	Pact with evil	Adam blames Carlson for burning down John's barn. Jeff accepts Adams' money. Jeff arrests Carlson
Conflict		Adam offers Mary to buy her farm. Mary accepts Adam's offer. John refuses to sell his farm. John gets angry with Mary
	Evil actions	Adam shoots John. John is injured
Outsider revealed		Mary witnesses Adam shooting John. Mary tells Jeff that Adam is a killer. John tells Jeff that Adam is a killer. Jeff gets angry with Adam
	Enlightenment	Jeff realizes that Carlson did not burn out John's barn. Jeff releases Carlson. Jeff says sorry to Carlson
Rise of the hero		John faces Adam. John demands Adam to leave. Adam refuses to leave the town
	Redemption	Jeff arrests Adam. Adam shoots Jeff. Carlson shoots Adam. Adam dies. Jeff dies
Conclusion		Mary says sorry to John. Carlson is freed

Table 7 Roles played by the character in the story in each of its subplots

	Destructive Outsider	Faust
John		Victim
Mary	Community Member	Victim
William	Community Member	
Jeff	Hero	Faust
Carlson		Victim
Adam	Outsider	Mephistopheles

reader, who by default compares with his own experience of short stories or plot summaries

To complicate matters, the present paper addresses the construction of multi-plot stories, which by their nature tend to be longer than single-plot stories generated by other automated storytellers. Prior attempts to address this problem have resorted to the development of quality metrics on the structural features of the output [16]. However, there is a shortage of meaningful metrics for plot, associated with the current lack of consensus on the definition of the term (as explained in Sect. 2.1). Under the circumstances, a qualitative rather than quantitative pilot evaluation has been carried out for the outputs of the solution proposed in this paper.

A set of 10 stories has been produced with a combination of the described weaving strategies. The distribution of weaving strategies and plot combinations for these stories are described in Tables 10 and 11. The actual stories discussed are presented in Appendix 8.

The analysis of system output presented in the remains of this section is intended as a formative rather than summative evaluation.

Juxtaposition Strategy

The plot for Story 1 (see Appendix 8) is generated by applying the juxtaposition strategy. This involves basically switching between plots at every scene, so that the sequence of scenes for the final story involves a continuous alternation between a scene from one subplot and a scene from the other. In this particular example, the approach works well, because both of the plot templates used ("Destructive Outsider" and "Faust") involve a character that misbehaves but whose evil actions are resolved towards the end of the story. The combination works fine, because the same character (Walter) has been chosen as protagonist for both subplots, and their nature aligns reasonable well as the story progresses. If a different character had been chosen to hold the central role in one of the subplots combined in this fashion, the result may have been less successful. On the other hand, given the choice of having the same character as central to both subplots, a different weaving strategy might have run the risk of breaking the alignment between the two processes of progressive downfall, thereby leading to less successful combinations.

Table 8 A sample story generated by combining three plots based on "Descent into the Underworld", "Creation of artificial life", and "Faust" by applying the "Chinese Boxes" strategy

Descent into Underworld	Creation of Life	Faust	Actions
Happy lovers			Hawa loves Seth. Seth loves Hawa. Korr hates Hawa. Korr hates Seth. Hawa has a magic sword. Hawa has a magic wand
Lost			Korr spells a cast against Seth. Seth is turned into stone
Mourning and quest			Hawa is sad. Hawa travels to Korr's kingdom
	Prometheus Dream		Hawa needs a minion
	Creation of new life		Hawa studies how to create life with magic. Hawa learns how to create life using dark magic. Hawa creates a minion using dark magic. Hawa names the minion Adam
	The being complies		Hawa orders Adam to fight a dragon. Adam defeats a dragon
	Rebellion and escape		Hawa orders Adam to fight a troll. Adam refuses to fight. Adam flies
	Infringement		The being infringes the human laws and moral principles
	Quest for the creation		Adam goes to a city. Hawa pursues Adam. Adam enters into Sam's store. Adam steals food and clothes. Sam discovers Adam. Adam kills Sam
	Duel with creation		Hawa finds Adam. Adam fights back Hawa. Hawa cast a spell on Adam. Adam disappears
Deal and brief reunion			Hawa arrives at Korr's castle. Hawa asks for a deal. Korr asks Hawa to give him her magic sword. Hawa accepts. Korr gives Hawa a potion to save Seth. Korr asks Hawa not to open the bottle before reaching Seth. Hawa travels back her country
Infringement			Hawa does not trust Korr. Hawa opens the potion before reaching Seth. The potion disappears
Metamorphosis			Hawa has lost Adam. Hawa is sad
		Frustration	Hawa is sad. Hawa regrets her life. Hawa studies dark magic
		Temptation	Hawa invokes an evil demon. Thrall appears. Thrall offers Hawa to be a queen
		Pact with evil	Hawa makes a deal with Thrall. Hawa becomes the queen
		Evil actions	Thrall persuades Hawa to raise the taxes. Hawa raise the taxes. Thrall persuades Hawa to punish the debtors. Hawa orders to imprison the debtors
		Enlightenment	Lem is a friend of Hawa. Lem tells Hawa that her vassals are suffering. Hawa realizes that she is acting badly
		Redemption	Hawa spells a cast on Thrall. Thrall curse Hawa. Thrall dies. Hawa dies

An important observation arising from this story is that, in some cases, two subplots that are similar in nature may align so well that the differences between them get blurred, especially if the same character is chosen as protagonist of each of them.

Random Strategy

The plots generated by means of random weaving (Stories 3 and 4 from Appendix 8) provide interesting examples of the different ways in which a random choice may impact the combination of two plots.

In story 3, Benson is the central character of a "Destructive Outsider" subplot and West is the central character in a "Creation of Artificial Life" subplot. The two subplots are combined at random, leading to an interesting distribution of the subplots over the whole of the story. The scenes of the "Destructive Outsider" subplot occupy the 1–2, 6–8, 10–11, and 13 spans of the story, and the scenes of the "Creation of Artificial Life" subplot occupy the 3–5, 9, and 12 spans of the story. This distribution is particularly successful, because it involves a leisured rhythm at the start of the story, where two or three scenes are devoted to introduce each subplot before moving onto the next. Subsequently, the rhythm of switching between subplots rises towards the end of the story. This happens to match well the nature of the subplots, which involve introductory descriptive scenes towards the beginning and more exciting events towards the end (revelation of the evil nature of Benson at 9, Benson being fired at 12, and Nemo's death at 13). Although this is a serendipitous achievement, it points to interesting heuristics that might be derived from such observations.

In story 4, Lupin is the central character of a "Destructive Outsider" subplot and Job is the central character in a "Creation of Artificial Life" subplot. Lupin acts as connecting character by also being the creature that Job creates. This leads to another interesting alignment, because the Lupin character shares in both subplots an evil tendency that becomes resolved (in different ways) towards the end. This actually makes it very tricky to identify which scene corresponds to which plot without checking the underlying system representation. Scenes 10–12 all involve Lupin misbehaving in various ways, and 10 is from the first plot and 11 and 12 from the second. There is another revealing feature that arises from the random combination strategy: Lupin is shot by Alain and dies in scene 14, and then discovered elsewhere by Job and also shot in scene 15. This is a result of weaving scenes from two different plots without checking consistency issues between them.

Introducing Precondition Satisfaction: Chinese Boxes

The stories generated with the Chinese Boxes strategy are stories S6, S7, S9, and S10.

Story 6 has a significant parallelism with story 4 described above, in as much as it also Lupin as the central character of a "Destructive Outsider" subplot and as the creature that Hubert creates in a "Creation of Artificial Life" subplot. This similarity

Table 9 Roles played by the character in the story in each of its subplots

	Descent into underworld	Creation of life	Faust
Hawa	Orpheus	Frankenstein	Faust
Seth	Eurydice		
Korr	Hades		
Adam		Creature	
Sam		Victim	
Thrall			Mephistopheles
Lem			Victim

Table 10 Distribution of the sample of stories for evaluation over the input parameters

	2 templates	3 templates
Random	2	–
Juxtaposition	1	–
Communicating vessels	3	1
Chinese boxes	2	3

Table 11 Reliance on specific plot templates for the stories evaluated

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10
Destructive outsider	X		X	X	X	X	X	X	X	X
Benevolent outsider		X					X	X	X	
Faust	X									X
Descent into underworld										
Creation of artificial life		X	X	X	X	X	X	X	X	X
Random			X	X						
Juxtaposition	X									
Communicating vessels		X			X			X		
Chinese boxes						X	X		X	X

has been engineered to allow for discussion of the differences arising from the different weaving strategies employed. The Chinese Boxes strategy inserts one subplot as a continuous sequence embedded in the other. In this particular case, the "Creation of Artificial Life" subplot occupies the 3–9 subspan of the full story, with the "Destructive Outsider" subplot covering the 1–2 and 10–16 spans. This might have lead to another inconsistency (if Lupin had been killed at the end of the "Creation of Artificial Life" subplot in scene 9), but the consistency checking procedure forces a different solution to the duel between the creature and its maker, resulting in the death of Hubert.

Story 7 constitutes an example of the combination of three subplots, also using the Chinese Boxes strategy. This leads to a story in which May as protagonist of a "Faust" subplot (span 2–7), Peggy as protagonist of a "Benevolent

Outsider" subplot (span 8–11), and Helen as protagonist of a "Destructive Outsider" subplot that acts as overarching setting for the other two. In this case, Helen acts as a connecting character, being helped by Peggy (the "Benevolent Outsider") to overcome her monetary difficulties and by May to face her "Destructive Outsider" Adam. The resulting story can be read as a tale of the misadventures of Helen, who is helped on her way by Peggy and May who have their own interesting background stories. This example helps to show the potential advantages of the use of combinations of subplots to construct interesting stories. It is very probable that the use of the Chinese Boxes strategy is instrumental to convey the impression that two of the plots are secondary to the other.

Story 9 constitutes an example of the combination of three subplots using the Chinese Boxes strategy, but with poor results. It is based on the by now well-known story of Hubert as protagonist of a "Creation of Artificial Life" subplot (spanning 3–9) with Lupin, the creature created by Hubert in the first subplot doubling up as protagonist of a "Destructive Outsider" subplot that acts as overarching setting. The story is now extended with an additional "Benevolent Outsider" subplot featuring Jack. The insertion is not very felicitous, with the subplot for Jack appearing as span 10–17 of the complete story, with little or no connection to the rest of the subplots other than taking place in the same town and sharing some secondary characters with them. This is an example of how addition of extra subplots with no connection to the rest of the story may decrease the quality of the story.

Story 10 constitutes an example of the combination of three subplots, also using the Chinese Boxes strategy. This leads to a story in which Hubert is the protagonist of a "Creation of Artificial Life" subplot (spanning scenes 3–6 and 13–14) and also the protagonist of a "Faust" subplot that is embedded within it (span 7–12). Lupin, the creature created by Hubert in the first subplot, doubles up as protagonist of a "Destructive Outsider" subplot that acts as overarching setting for the other two. This example shows the result of nesting down to two levels. In this particular case, as already mentioned for some of the earlier examples, the alignment between subplots leads to a successful combination, with the character of Hubert combining the role of creator of unnatural creatures (in the "Creation of Artificial Life" subplot and of making pact with evil in the "Faust" subplot), meeting his deserved end at the end of the "Creation of Artificial Life" subplot (at the hands of the creature he created). There is a slightly less successful combination in the character of Lupin, which plays the role of evil character both in the "Creation of Artificial Life" subplot and the "Destructive Outsider" subplot. The alignment here is less felicitous, because it involves a mismatch: Lupin is introduced as a stranger arriving to town in scene 2 (behaving much like any human would), but then created from a wolf using magic by Hubert in scene 4 (and then taught to behave like a human in scene 5). This type of inconsistency arises in the current version of the system due to the fact that precondition satisfaction is only checked against the immediately neighbouring scene in each case, and, at the connection point, scene 3 (from the "Creation of Artificial Life" subplot) does not mention Lupin, and the inconsistency arises between

scenes 2 and 4, which are separated by scene 3. This problem will be addressed in further work.

Using Precondition Satisfaction to Drive the Weaving: Communicating Vessels

The stories generated with the Communicating Vessels strategy are stories are S2, S5, and S8.

Story 2 uses the Communicating Vessels strategy to combine an instance of the "Benevolent Outsider" subplot with Ernest as the central character with an instance of the "Creation of Artificial Life" subplot that has Alexander as main character. As the Communicating Vessels strategy combines elements from the two plots at different rhythms, the spans covered by each subplot are variable in their distribution: spans 1–5, 7–8, and 12 for the "Creation of Artificial Life" subplot and spans 6 and 9–11 for the "Benevolent Outsider" subplot. In this case, the "Benevolent Outsider" subplot acts as a secondary plot, with Ernest helping Alexander solve the problem presented by the creature he has created. The scenes for the subplot are inserted reasonably among those for the main plot, and there appear to be no obvious inconsistencies.

Story 5 combines Benson as the "Destructive Outsider" with West's effort at "Creation of Artificial Life". In this case, Mary acts as the connecting character, featuring as victim of the villains in both subplots, Benson as the "Destructive Outsider" and Nemo as the creature created by West. The distribution of spans is 2, 4–5, and 7–8 for the "Destructive Outsider" and 1, 3, 6, and 9–12 for the "Creation of Artificial Life" subplot. The subplots are combined with more presence of the "Destructive Outsider" at the start of the story, with elements from "Creation of Artificial Life" inserted sparsely, and a final stretch of the story that focuses entirely on the "Creation of Artificial Life" subplot. There is an effect of this initial "Destructive Outsider" subplot building the reader's sympathy for Mary (as survivor of the depredations of Benson) only to heighten the impact of her death at the hands of Nemo. Consideration of such effects may be a valuable heuristic in future versions of the system.

Story 8 constitutes an example of the combination of three subplots, but it uses the Communicating Vessels strategy instead of the Chinese Boxes strategy. It involves a story in which Benson features again as the "Destructive Outsider" and West as the creator in an 'Creation of Artificial Life" subplot, but with the addition of a "Benevolent Outsider", Hugo, who arrives at the point in the story where Mary has died and sacrifices himself to save her by magically resuscitating her. The distribution of spans in this case is 2, 4–5, 7–8 for the "Destructive Outsider", 1, 3, 6, and 9–11 for the "Creation of Artificial Life" subplot, and 12–17 for the "Benevolent Outsider" subplot. In this particular case, the weaving of the additional plot into the story has not led to any considerable improvement to the quality of the story. This can stand as a good example of how an increase in the number of subplots in a story needs to be justified by some valuable synergy between the new subplot and the rest of the story. Future work will consider the identification of heuristics to inform such processes.

Summary of Evaluation Results

The results of the formative evaluation are summarised in Table 12, which lists the plot templates combined for each strategy, the weaving geometries that resulted, and a summary of whether the resulting story produced a positive impression or a negative impression according to the analysis presented above.

The negative impressions and the features in the story that lead to them are summarised here:

- Stories 4 and 10 involve characters appearing in the story either before being born or after having been killed. This is a result of not checking preconditions in the Random strategy used for Story 4, but derives from restricting the checks to boundary scenes between subplots in the Communicating Vessels strategy used in Story 10.
- Stories 8 and 9 involve subplots added to the story that do not improve the story, because they are not well related to the rest of the material. This is a problem for both the Chinese Boxes (used for story 9) and Communicating Vessels (Story 8) strategies.

The features that have lead to positive impressions during the formative evaluation are summarised in the Sect. 6.1.

Discussion

This section discusses insights arising from the formative evaluation, issues of the developed system, both in terms of features that require attention in their current form and in terms of comparison with previous work.

Insights Arising from the Formative Evaluation

Each of the stories considered has been analyzed to identify relevant characteristics that might have impact on the perception of its relative quality by potential readers. Although it is clear that most of the positive characteristics appear as the result of serendipity, they can be considered as potentially useful features to address in future versions of the system. Negative characteristics observed in the stories arise from problems in the current operation of the system that may be refined for later versions.

The following insights have been considered relevant:

1. When preconditions are not considered in the weaving procedure, inconsistencies arise (Story 4).

2. Even when preconditions are considered, the current solution sometimes results in inconsistencies that span beyond the two scenes at a contact point between two plots (Story 10).
3. Alignment of protagonists for different subplots works well when there is a certain conceptual alignment between the plots and the plots are interwoven tightly (see Stories 1, 4, and 10).
4. Frequency of transition between plots conveys an impression of rhythm to the story, with frequent changes resulting in stories with faster rhythm and less frequent changes in stories with slower rhythm (as in Story 3).
5. The adequacy of this "rhythm" of a story at a particular point may be related to the descriptive vs. narrative nature of the scenes at that point (Story 3)
6. Imbalance between subplots—for instance by centering one more towards the beginning and one more towards the end of the story—can be used to induce particular effects, such as using one plot to set up a situation that heightens the effect of the second plot (see Stories 3 and 5).
7. The emotional impact of each subplot on the reader may be a valuable consideration to decide on its relative placing with respect to other subplots (Story 5)
8. Subordination between plots may be used to advantage (as in Story 7).
9. Some plots fit in well as ancillary contributions to a larger story (Stories 2 and 7).
10. Such ancillary subplots work well both when using the Chinese Boxes strategy (Story 7) and the Communicating Vessels strategy (Story 7)—provided that they are inserted at the right point of the story. For this purpose, it is important to consider what the interaction between the actions in the two subplots are.
11. Some subplots may act as stitching for a set of subplots, with their main characters appearing as secondary characters in the other subplots (Story 5).
12. Subplots inserted in a story may not be a valuable contribution unless they share some relevant characters—as established by the roles of the corresponding plot templates—with the subplots already in the story (Story 9).
13. Even when it involves the same characters of a story, an additional subplot may not be a valuable addition if it links up with the story at a point where the other subplots have been resolved already (Story 8).

Issues Requiring Attention

An important limitation of the proposed solution arises from the abstract nature of the representation of plot that is being considered. Although the Plot Weaver checks for consistency of preconditions and postconditions of contiguous scenes, inconsistencies can arise at a global level. As discussed in the definitions of plot template and precondition satisfaction, a plot template outlines necessary conditions for a story to be considered to feature that plot template. However, the only limits it establishes on what else the story might contain are those arising from the preconditions of the scenes in it. This allows for different plot templates to be woven together. It also requires that once the plot structure for a story is fixed, a process of detailed enrichment be applied to fill in any gap in consistency remaining in the story. This is a

Table 12 Summary of weaving strategies and plot templates used in the evaluation, the geometries of resulting combinations for the different strategies over test outputs, and overall appraisal of the resulting stories. Every asterisk marks from which story template has been taken the scene

Str	#	Plots																										Works
Alternating	1	DestOut	*		*		*		*		*		*															✓
		Faust		*		*		*		*		*																
Random	3	DestOut	*	*				*	*	*		*	*		*													✓
		CreaLife			*	*	*				*			*														
Random	4	DestOut	*	*		*		*	*		*	*					*			*								✗
		CreaLife			*		*			*			*	*	*			*										
CommVessels	2	BenOut						*			*	*	*															✓
		CreaLife	*	*	*	*	*		*	*				*														
CommVessels	5	DestOut		*		*	*		*	*																		✓
		CreaLife	*		*			*			*	*	*	*														
ChineseBox	6	DestOut	*	*								*	*	*	*	*	*	*	*									✓
		CreaLife			*	*	*	*	*	*	*																	
CommVessels	8	DestOut		*		*	*		*	*																		✗
		BenOut												*	*	*	*	*	*									
		CreaLife	*		*			*			*	*	*															
ChineseBox	7	DestOut	*											*	*	*	*	*	*	*								✓
		BenOut								*	*	*	*	*														
		Faust		*	*	*	*	*	*																			
ChineseBox	9	DestOut	*	*																	*	*	*	*				✗
		BenOut									*	*	*	*	*	*	*	*										
		CreaLife			*	*	*	*	*	*																		
ChineseBox	10	DestOut	*	*										*	*													✗
		Faust							*	*	*	*	*															
		CreaLife			*	*	*	*																				

process already operating in the construction procedure for single-plot stories, and described in [8]. However, the solution adopted in the present version of the system to make this freedom possible (restricting the checking of preconditions for scenes to the point of contact between two plot templates in a story) has some unwelcome consequences. A significant case that can easily occur in the current model is the reappearance of a character killed in a scene of one plot in later scenes of the other plot, creating a blocking inconsistency. Example of this type of situation was discussed for Story 4 as analyzed above (featuring the repeated death of a character) and Story 10 (featuring the birth of character that had already been active in the story at an earlier point). This situation can be amended by extending the consideration of precondition satisfaction to fuller contexts including the whole span of preceding plot lines. To implement such an approach without compromising the freedom of combination of plot templates is under consideration for future work.

The Plot Weaver supports the application of different weaving strategies. In its current version, the choice of strategy is determined by an input parameter provided by the user. Later versions of the system may consider alternative approaches. The space of possible stories is extremely large if the full capabilities of the underlying storytelling system are put into play (detailed construction of scenes and natural language generation for construction of elaborate texts).

However, this would very likely cloud the issue of the relative merits of different weaving strategies. If procedures are added to render the stories in fluent text, this might make it difficult for a reader to work out whether particular merits in a given solution arise from a specially successful weaving strategy or from better than average construction of the individual scenes involved. To reduce this difficulty, we have focused the material in this paper on the aspects of plot line weaving. A small subset of similar scenes is used throughout so as to allow the reader easy comparison across stories to highlight the effect of different weaving strategies. The proposed solution as it stands does not currently consider metrics for quality [19, 32] of particular weavings, though such a feature should be given high priority in future work.

The plot weaving techniques presented so far have proven to be feasible from a computational point of view. The comparison with the baselines has proven inconclusive, with good and bad weavings resulting from both baselines and literary approaches. This suggests that additional criteria may be required. After all, writing good human-like stories entails a good number of validations in terms of knowledge representation and consistency.

The insights outlined in Sect. 6.1 constitute a useful starting point for the development of knowledge-based heuristics that may yield a substantial improvement in the perceived quality of system outputs.

The solution presented in our paper is not intended to be exhaustive with respect to the set of possible strategies for the combination of plot lines. The design of appropriate computational solutions must consider requirements from both human approaches and computational restrictions. In the present paper, a selection of strategies reported by researchers studying literary works [23] has been considered in terms of their computational feasibility and the perceived quality of the resulting combinations. A review of all possible strategies used by professional writers would require a specific survey paper just for that purpose, and is beyond the scope of the

Table 13 Description of the plot templates employed

Template	The destructive outsider			The benevolent outsider		
Roles	`Outsider`			`Outsider`		
	`Hero`			`Community members`		
	`Community members`			`Enemy`(opt)		
Scenes	1	*Peaceful community*	`Community members` introduced	1	*Crisis*	`Community members` in crisis
	2	*Arrival*	`Outsider` arrives to the community	2	*Arrival.*	`Outsider` arrives to the community
	3	*Outsider destructive actions*	`Outsider` acts undiscovered against`Community members`	3	*Outsider helping actions*	`Outsider` tries to help-`Community members`
	4	*Outsider revealed*	Evil nature of `Outsider` revealed	4	*Outsider revealed*	`Outsider` revealed as the saviour
	5	*Rise of the hero*	`Hero` rises and fights against `Outsider`	5	*Sacrifice*	`Outsider` sacrifices herself/himself for the community
	6	*Purge*	`Outsider` purged. `Community Members` peaceful again			

Table 13 (continued)

Template	Faust			Descent into underworld		
Roles	`Faust`			`Orpheus`		
	`Mephistopheles`			`Eurydice`		
	`Victims`			`Hades`		
Scenes	1	*Frustration*	`Faust` regrets his/her fruitless life	1	*Happy lovers*	`Orpheus` and `Eurydice` are happy lovers.
	2	*Temptation*	`Faust` tempted by `Mephistopheles`	2	*Lost*	`Eurydice` is lost to `Orpheus`
	3	*Pact with evil*	`Faust` agrees to serve `Mephistopheles`	3	*Mourning and quest*	`Orpheus`mourns `Eurydice` and searches for her
	4	*Evil actions*	`Faust` performs evil actions	4	*Deal and brief reunion*	`Orpheus` makes deal with `Hades`
	5	*Enlightenment*	`Faust` becomes aware of being enslaved by evil	5	*Infringement*	`Orpheus` breaks rule and looses `Eurydice`
	6	*Redemption*	`Faust` performs saving action and dies defeating evil	6	*Metamorphosis*	`Orpheus` develops sublime art

Table 13 (continued)

Template	Creation of artificial life		
Roles	`Frankenstein`		
	`Creature`		
	`Victim`		
Scenes	1	*Prometheus Dream*	`Frankenstein` dreams about creating life
	2	*Creation of new life*	`Frankenstein` gives life to `Creature`
	3	*Creature complies*	`Creature` obeys `Frankenstein`
	4	*Rebellion and escape*	`Creature` rebels against `Frankenstein`
	5	*Infringement.*	`Creature` misbehaves
	6	*Quest for creature*	`Frankenstein` searches for `Creature`
	7	*Duel with creature*	`Frankenstein` fights `Creature`

present paper. To illustrate the computational aspect, a number of baseline strategies—arising solely from computational principles—have been included for comparison with the chosen human-inspired solutions. There are surely additional possibilities that might be considered, which we hope to address in future work.

Comparison with Prior Work

The proposed solution includes two strategies (Communicating Vessels and Chines Boxes) that take into account precondition satisfaction for plot elements before they can be inserted into a story draft. This procedure resembles the chaining of planning operators based on precondition satisfaction as used by the planning-based approaches to narrative discussed in Sect. 2.1. However, there is a significant difference between the two approaches in as much as the planning approach relies mainly on the precondition satisfaction relation to structure a story. In contrast, the proposed solution relies mainly on the structure of the subplot lines arising from different templates, and only relies on the precondition satisfaction to validate shifts across different plot lines. Another significant difference is that the satisfaction of preconditions in planning systems requires that the precondition be explicitly present in the context to be satisfied, whereas the type of satisfaction considered in the proposed solution is focused more on consistency between preconditions and the conditions holding in the context. In terms of traditional precondition satisfaction, this corresponds to a situation that allows for preconditions that are simply missing from the context to be accommodated [21] if they do not conflict with prevailing conditions. This difference is fundamental to allow the freedom of combination of templates required by the proposed method.

A comparison with existing solutions for the combination of more than one subplot into multiple-plot stories faces an important obstacle in that different approaches are based on different understandings of what the instances of "plot" to consider are. A discussion of the main differences follows.

The solution for combining subplots into multiple-plot stories proposed in this paper has much in common with the solution presented by Fay in [12]. In both cases, there is an algorithm that considers a number of subplots, each one already built to a conclusion, and combines them into a story by interleaving them in some manner, allowing for the possibility of major characters in one subplot being bound to lesser characters in another. The solution advocated by Fay differs from the present one on a number of aspects. First in that, it operates over instances of "plot" that represent narrative threads focalised on a particular character, each of them allowing participation by other characters which are represented by "generic entities". In this way, individual threads are not actually complete plots, but rather partial views of the contribution of a particular character to one of the full stories that constitute the reference corpus for the system.

Second, in that the algorithm proposed by Fay combines processes of interleaving plot elements from different threads with processes of fusing plot elements by binding the main character of one with some "generic entity" in the other. In the first case, elements from different threads are added as different elements for the

final story. In the second case, two plot elements from different threads contribute a single-fused plot element to the final story.[2] Fay's solution, therefore, operates at a lower level of granularity in terms of its conception of plot: the elements that it combines are units of abstraction arising not so much from their relevance to the development of the plot but rather from being restricted to the contribution of a particular character.[3] This allows for more emergent behaviour in the resulting stories (as discussed by Fay), but makes the solution less suitable for combining actual stories. The solution proposed in the paper addresses a slightly different problem which involves combining full-blown plots, each of them corresponding to a complete story in its own right.

Porteous et al. [26] describe a solution that relies on an understanding of plot closer to the one used in this paper, as it is designed to combine separate storylines in which a main character has a particular goal that it has to achieve to close the storyline successfully. It also includes the option of binding major characters in one subplot to lesser characters in another. It differs significantly from both Fay's solution and the present proposal in that it does not consider a set of already specified plots to be combined, but rather relies on a planning-based approach that incrementally builds each of the plots from planning operators to match the requirements of the combined story at each point. This difference is important for the stated purpose of their solution, which is to drive an interactive narrative experience, and which requires that the story be at each point tailored to match (and react to) any contributions by the user participating in the experience. As was the case for Fay's method, this makes this solution unsuitable for combining plots that are already fully structured.

The problem facing the Raconteur system [15] differs considerably from the one addressed in the present paper with respect to the constraints that the inputs under consideration impose on the process. The Raconteur system operates from a set of predicates describing the moves in a chess game. Each predicate describes a move taking place at a particular location within the chess board at a particular turn of the game. Choices of how to tell a story about the game are constrained by the relative chronological order between the moves selected for inclusion in the story, and by the need for pieces participating in a move as characters to be reasonably close to one another for an interaction between them to be believable. For these reasons, temporal and spatial constraints are paramount to the process, and they take priority when deciding how to weave the resulting narrative threads, making co-occurrence of suitable characters at particular location and temporal coherence over the scenes in a story fundamental to the decision process. In contrast, in the solution proposed here, there is no prior map that fixes the relative distance between locations mentioned in different plot templates, so the act of weaving two plots together in a particular way can actually determine that the locations mentioned in them become spatially close as required to make the implied story world coherent. In a similar fashion, because

[2] This corresponds to the process of *scene merging* mentioned above which is not currently considered in our system, but will be addressed in future work as a possible extension to the system.

[3] This point is taken up again below in the discussion of differences with the StoryFire system.

there are no absolute time stamps for the scenes in each plot template, particular weaving patterns will impose on the implied story world a partial chronological order between the corresponding scenes.

The StoryFire system [17] combines the task of telling a known set of events the way it happened with the goal of ensuring the tale follows an interesting plot line. This is achieved by sometimes omitting known events (if they are not relevant to the desired plot) and sometimes inserting fictional events (if they are needed for the plot to make sense). This approach that mixes reality and fiction in a flexible way is closer to the way humans construct stories than that of many other computational storytellers. The concept of plot in StoryFire is very similar to the concept of plot template proposed in the present paper, including both a sequence of scenes and a set of roles, yet the system handles the additional level of *narrative threads*, which represent the view of the story world that a particular character has. These are closer to the plot lines considered by Fay, as discussed above, and the narrative threads of the Raconteur system—on which they are based. The fact that different systems consider these two distinct concepts of a linear representation of the content of a story, each holding a partial view based on different criteria, suggests that a more elaborate representation of the nature of stories is required for a fruitful solution of storytelling that mixed reality and fiction. Under this view, stories would not be a simple woven fabric of threads of one or the other kind, but rather a complex fabric where threads of two different types (plot lines and focalised narrative threads) come together in a very complex weave. The interplay between these two different views of a story has already been discussed in the section 3.2 related to precondition satisfaction. Intuitively it may seem that, because different plot lines will tend to be focalised differently (following a different character), shifts between subplots will usually imply changes in focalisation, whereas simple shifts in focalisation occur more often within a single plot (for instance, as the story traces the activity of the Maker and the Creature as they move separately about the world in a "Creation of Artificial Life" plot). However, some of the examples of generated stories analyzed in Sect. 5 (see for instance, Story 4) suggest that the weaving of plot templates may result in the juxtaposition of scenes from different plot templates with no intervening change in focalisation, in particular when relevant roles of two different subplots have been bound to the same character. The solution presented in this paper focuses specifically on the combination of plot lines rather than combination of focalised narrative threads. Nevertheless, the interaction between these two different types of weaving needs to be explored in further work.

The corpus-based solution applied by the PlotAssembler system [18] for the construction of plots highlights the importance of character co-occurrence across these shifts in focalisation and/or plot lines. Almost all of the systems described above include a process either for validating bindings between characters from different plots (when characters in plots are described generically and therefore may be assigned to refer to entities appearing in different plots) or for identifying characters that co-occur across different threads (which suggests a potential value for using the character to lead from one to the other during a shift in focalisation). The small spans of plot used as units by PlotAssembler also include explicit representation of the roles played in them by characters. The use of probabilities of co-occurrence in a

corpus is a potentially valuable source of additional information that may considered to enhance decision processes in further work.

The present paper focuses on the problem of how full plots may be used as subplots to be woven into larger stories. To address this issue experimentally, we relied on an existing storytelling system: INES [8, 9], which provides a broad number of functionalities such as elaborate knowledge representation for characters and actions, procedures for enriching abstract descriptions of scenes into more elaborate constructions, and methods for rendering the results as natural language text. It also relies on a software architecture based on micro-services that allows a number of modules addressing these various functionalities to collaborate in an orchestrated fashion. However, a detailed description of these features and the associated functionalities would be beyond the scope of the present paper and would be likely to cloud the issue of plot weaving. For this reason, the present paper focuses on a smaller set of simpler concepts that are relevant to the issue of multiple-plot integration, attempting to avoid terminological distinctions that are important for other functionalities of the underlying storytelling system but not necessary to understand the issues being described here. This is achieved by rephrasing some of the technical aspects in terms specifically relevant to the plot weaving issue, and avoiding issues arising from other functionalities of the underlying storytelling system. Readers interested in the details concerning other functionalities of the storytelling system are invited to consult prior publications on the INES system (on architecture [6, 7], on knowledge representation [5, 10], and on generation of simple plots [8]).

Conclusions and Future Work

The exploration undertaken in this paper has identified a set of concepts relevant to the task of plot weaving, established some computational baselines for strategies for weaving plots, and uncovered relevant intuitions as to the impact of particular types of weaving on the impression they produce on a reader. The baseline weaving techniques outlined in Sect. 4 have demonstrated their computational feasibility and relative merits as outlined in Sects. 5 and 6.

The two advanced plot weaving techniques proposed—Communicating Vessels and Chinese Boxes—come from the literary context, proving the feasibility of translating these creativity techniques from human-made literature to automatic storytelling. Although the algorithms have been developed for a particular target system, the conceptual model in which they are based is sufficiently generic to be exported to other storytelling systems.

The consistency checking mechanism presented, even though there is still room in it for improvements, helps to provide an reasonable continuation between scenes from different plot lines.

The use of a random weaving strategy as a baseline has also been revealed as a very interesting tool for comparing the quality of other weaving techniques. Contrary to expectations, there are no substantial differences in perceived quality of the resulting stories between the random strategy, the mechanical juxtaposition strategy, and the more elaborate techniques inspired in literary studies. In all cases, there

were both positive and negative examples among the generated output. The random strategy is therefore a reasonable baseline that sets a threshold that other strategies will have to transcend. Future research efforts on performing a user-oriented survey for analyzing the quality of the stories will probably use this baseline.

The insights listed in Sect. 6.1 constitute a valuable starting point for the development of further heuristics to drive plot weaving procedures. An important conclusion from the present study is that such heuristics will need to have a substantial knowledge-based component to address the perceived importance of alignment between and interaction across different subplots within a story.

Main Contributions

The main contributions of the present paper to the research effort into the weaving of multiple-plot lines can be summarise as follows.

If stories are to be built in terms of plot-relevant elements, these need to be represented at a level of abstraction higher than the traditional story representation techniques. This implies that plot elements need to be slightly more generic than a full conceptual description of a scene, and that the details need to come from an additional process of instantiating a plot element into a detailed scene. The concept of plot template as presented in the paper—as an abstract unit of representation for elements of narrative structure that can be combined with others of the kind—is a valuable contribution to the literature.

The roles of characters relative to specific subplots play an important role in the decision processes for plot weaving.

The proposed procedure for checking consistency between subplots at points of transition between subplots in the discourse of the story constitutes a valid baseline that does not compromise the freedom of representation required for plot descriptions.

Future Work

Throughout the paper, a number of relevant tasks have been identified as potential future work.

The current solution based on scene weaving can be extended with functionality for scene merging (see Sect. 3.2).

Solutions need to be considered for the extension of preconditions satisfaction to the whole span of preceding plot lines without compromising the freedom of combination of plot templates (see Sect. 6.2).

Alternative solutions must be found for deciding which strategy to apply (see Sect. 6.2).

Knowledge-based heuristics or metrics of quality must be developed to decide when the addition of further plots is a positive contribution to a story or when one weaving is better than another (see Sects. 6.1 and 6.2).

Further alternatives for weaving strategies need to be designed, either based on those used by professional writers or on additional computational methods.

Acknowledgements This paper has been partially funded by the projects *CANTOR: Composición Automática de Narrativas personales como apoyo a Terapia Ocupacional basada en Reminiscencia*, MICINN Grant. No. PID2019-108927RB-I00 and *InVITAR-IA: Infraestructuras para la Visibilización, Integración y Transferencia de Aplicaciones y Resultados de Inteligencia Artificial*, UCM Grant. No. FEI-EU-17-23.

A Examples of Generated Stories

This appendix lists the textual rendering of the stories used for the evaluation presented in Sect. 5. For the sake of concision, this rendering has exclusively included the key plot actions.

Story 1 combines plots based on "Destructive Outsider" and "Faust" plot templates using the Alternating Juxtaposition weaving strategy:

[DO1] Becky, Phil, and William work in a logistics company. [F1] Walter is a businessman frustrated, because his company went bankrupt. [DO2] Walter becomes the new CIO in a logistics company. Phil welcomes Walter. Phil introduces Walter to Becky and William. [F2] Alan is a tradesman. Alan offers Walter to participate in shady business. Alan offers Walter 1 million dollars for participate. [DO3] Walter commits fraud in his job. Walter blames William for the fraud. Walter fires William. Becky believes Walter. Phil does not believe Walter. [F3] Walter accepts Alan offer. [DO4] Phil investigates William's laptop. Phil discovers that Walter committed fraud. Phil discovers that William is innocent. [F4] Walter uses his job for money laundering. Walter hides his actions. [DO5] Phil tells Becky that Walter is a criminal. Phil and Becky talk with Tax Office to report Walter's fraud. [F5] Walter finds out that Alan works for a terrorist group. Walter feels miserable. [DO6] Walter is fired for fraud. William is readmitted. Phil and Becky welcome back William. [F6] Walter reports the police that Alan works with terrorists. Alan shoots Walter.

Story 2 combines plots based on "Benevolent Outsider" and "Creation of life" plot templates using the Communicating Vessels weaving strategy:

[CL1] Alexander's wife died recently. Alexander dreams of bringing to life his wife. [CL2] Alexander discovers a spell to bring to life a portrait of his wife. [CL3] Alexander talks every evening with the portrait. [CL4] Alexander falls in love with Mina. Alexander tells the portrait he loves Mina. [BO1] Alexander receives the visit of Ernest. [BO2] Alexander allows Ernest to stay in the guest house. [CL5] The portrait is jealous of Mina; the portrait takes Mina and disappears. [CL6] Alexander looks for Mina across the castle. [BO3] Ernest helps Alexander to find the picture by means of the magic. [BO4] Ernest reveals that he is a miracle worker who helped Alexander's family years ago. [BO5] The picture attacks Alexander with magic.

Ernest saves Alexander by sacrificing himself. [CL7] The picture tells Alexander that he must defeat it to save Mina. Alexander burns the portrait. Alexander and Mina are happy.

Story 3 combines plots based on "Destructive Outsider" and "Creation of life" plot templates using the Random weaving strategy:

[CL1] West is a scientist. Lily is West's wife. Mary is West's sister. West lives with his family in the countryside. [DO1] West is a wealthy man; Mary loves West; West loves Mary. [CL2] West dreams with creating artificially a boy. [DO2] Benson is the new steward of West Manor. [DO3] Mary falls in love with Benson. [DO4] Benson seduces Mary. [CL3] West creates artificially a boy. West names the boy as Nemo. [CL4] West teaches Nemo to behave as an ordinary boy. [CL5] Nemo accidentally kills Mary. Nemo discovers that he is not human. Nemo leaves home. [DO5] Benson is discovered by Mary while stealing money. [CL6] Nemo steals food in a farm. [CL7] West looks for Nemo. [DO6] West fires Benson. Mary is happy again. [CL8] West finds Nemo in an abandoned house in the town. West deactivates Nemo. West is sad and sorry.

Story 4 combines plots based on "Destructive Outsider" and "Creation of life" plot templates using the Random weaving strategy:

[DO1] Augustine is a farmer. Claude is the major of the town. Alain and Peirce are forest wardens. [DO2] Lupin is an foreigner that arrives to the town. Claude welcomes Lupin. Lupin hires a room in Augustine's farm. [CL1] Job is a magician. Job wants to create humans from animals. [DO3] Claude visits Augustine. Augustine tells Claude that Lupin is very helpful. [CL2] Job captures a wolf and transforms it into a human. Job names the new creature as Lupin. [CL3] Lupin becomes a werewolf during the night; Lupin kills Augustine. [DO4] Lupin tells Claude that a wolf attacked Augustine. Claude asks Alain to find and kill the wolf. [CL4] Job teaches Lupin how to behave like a human; Lupin thanks Job for being his master. [DO5] Alain looks for the wolf in the forest. Alain asks Pearce for help. [DO6] Lupin becomes a werewolf during the night. Lupin attacks Pearce and Alain. [CL5] Lupin becomes a werewolf during the night. Lupin realizes that he is not a human. Lupin escapes from Job's house. [CL6] Lupin attacks a traveler; Lupin steals traveler's clothes. [CL7] Job follows Lupin's trace across the country. [DO7] Alain shoots Lupin. Lupin dies. [CL8] Job discovers Lupin near a farm. Job shoots Lupin. Lupin dies. Job goes back home. [DO8] Pearce and Alain become heroes in the town.

Story 5 combines plots based on "Destructive Outsider" and "Creation of life" plot templates using the Communicating Vessels weaving strategy:

[DO1] West and Mary live in a mansion. [DO2] Benson is the new steward of West Manor. [CL1] West wants to create a robot. [DO3] Mary falls in love with Benson. [DO4] Benson seduces Mary. [CL2] West creates robot imitating a boy. West names the boy as Nemo. [DO5] Benson is discovered by Mary while stealing money. [DO6] West fires Benson. [CL3] West introduces Nemo to Mary. [CL4] Mary plays with Nemo in the garden. Nemo accidentally kills Mary. Nemo leaves the house. [CL5] West discovers Mary's corpse. West is devastated. West looks for Nemo in the town. [CL6] West finds Nemo in the town. West deactivates Nemo. West kills himself.

Story 6 combines plots based on "Destructive Outsider" and "Creation of life" plot templates using the Chinese Boxes weaving strategy:

[DO1] Augustine is a innkeeper. Claude is the major of the town. Alain and Pearce are hunters. [DO2] Lupin is an foreigner that arrives to the town. Claude tells Lupin to lodge in Augustine's inn. [CL1] Hubert is a magician. Hubert wants to create humans from animals. [CL2] Hubert captures a wolf and transforms it into a human. Hubert names the new creature as Lupin. [CL3] Hubert teaches Lupin how to behave like a human. [CL4] Lupin becomes a werewolf during the night. Lupin escapes from Hubert's house. [CL5] Lupin goes to the forest; Lupin kills a deer. [CL6] Hubert follows Lupin's trail. [CL7] Hubert discovers Lupin near a farm. Lupin kills Hubert. [DO3] Lupin is very kind with Augustine. Augustine tells Claude that Lupin is very helpful. [DO4] Lupin becomes a werewolf during the night; Lupin kills Augustine. [DO5] Lupin tells Claude that a wolf attacked Augustine. [DO6] Alain asks Pearce for help. [DO7] Lupin becomes a werewolf during the night. Lupin attacks Pearce and Alain. [DO8] Alain shots Lupin. Lupin dies. [DO9] Pearce and Alain become heroes in the town.

Story 7 combines plots based on "Destructive Outsider", "Faust", and "Benevolent Outsider" plot templates using the Chinese Boxes weaving strategy:

[DO1] Helen lives in May's house. [F1] May is frustrated with her life; May has lost her job; May has no money. [F2] Jeff is the manager of a discotheque; May meets Jeff in his discotheque; Jeff offers May a shady business at the discotheque; May accepts Jeff's offer. [F3] Jeff asks May to sell drugs in the discotheque; May starts selling drugs in the discotheque; Jeff gives money to May; May becomes rich. [F4] May takes drugs; May gets sick; [F5] May realizes that drugs are bad. [F6] May quits Jeff's job; May calls the police; Jeff shoots May; May go into a coma; Police arrests Jeff; May is taken into hospital. [BO1] May has brain damage; May is in the hospital; Helen is very worried about May; Helen has run out of money. [BO2] Peggy is an old lady; Peggy is the new neighbour of Helen; Helen welcomes Peggy; Helen is distressed. [BO3] Peggy asks Helen about her problems; Helen tells Peggy she has no money and May needs medicines; Peggy gives some money to Helen; Helen thanks Peggy. [BO4] Peggy reveals Helen that she is May's mother; Peggy gives all her savings to Helen; Helen pays May's treatment with the money. [BO5] May gets well; Peggy is very happy; May is very happy; Helen is happy. [DO2] Helen meets Adam; Adam is a guitar player; Adam is very kind with Helen. [DO3] Adam seduces Helen; Helen falls in love with Adam. [DO4] Adam asks Helen to join his band; Helen joins Adam's band; Helen is happy. [DO5] Adam drinks alcohol; Adam hits Helen; Adam abuses Helen. [DO6] Helen tells May that Adam is bad; May tells Helen to leave Adam; May no longer loves Adam. [DO7] May faces Adam; May leaves Adam's band. [DO8] May and Helen move to a new apartment; May and Helen live together.

Story 8 combines plots based on "Destructive Outsider", "Benevolent Outsider" and "Creation of life" plot templates using the Communicating Vessels weaving strategy:

[DO1] West is a magician. West and Mary live in a castle. West has a hidden treasure. [DO2] Benson is the new steward of West castle. [CL1] West wants to create a human being. West studies dark magic. [DO3] Mary falls in love with Benson.

Benson wants the hidden treasure. [DO4] Benson seduces Mary. [CL2] West creates a creature imitating a boy. West names the boy as Nemo. [DO5] Benson is discovered by Mary while stealing the treasure. [DO6] West imprisons Benson. [CL3] West introduces Nemo to Mary. [CL4] Mary plays with Nemo. Nemo accidentally kills Mary. Nemo leaves the house. [CL5] West is devastated. West decides to keep Mary's corpse in a case. [BO1] Hugo visits West's castle. [CL6] West finds Nemo in the forest. West turns Nemo into stone. [BO2] Hugo offers help to West. [BO3] Hugo reveals West that he is Mary's father. [BO4] Hugo casts a spell to revive Mary. Hugo sacrifices himself for Mary. [BO5] Mary is happy. West is happy.

Story 9 combines plots based on "Destructive Outsider", "Benevolent Outsider" and "Creation of life" plot templates using the Chinese Boxes weaving strategy:

[DO1] Augustine is a innkeeper. Claude is the major of the town. [DO2] Lupin is an foreigner that arrives to the town. Lupin hires a room in Augustine's inn. [CL1] Hubert is a magician. Hubert wants to create humans from animals. [CL2] Hubert captures a wolf and transforms it into a human. Hubert names the new creature as Lupin. [CL3] Hubert teaches Lupin how to behave like a human. [CL4] Lupin becomes a werewolf during the night. [CL4] Lupin goes to the forest. Lupin kills a deer. [CL5] Hubert follows Lupin's trail. [CL6] Hubert discovers Lupin near a farm. Lupin kills Hubert. [DO3] Lupin is very kind with Augustine. [DO4] Lupin becomes a werewolf during the night. Lupin kills Augustine. [DO5] Lupin tells Claude that a wolf attacked Augustine. [BO1] Claude is sick. Jack arrives in town. [BO2] Jack tells Claude that the town is in danger. Claude does not trust Jack. [BO3] Jack reveals that there is a cursed stone in the town. The curse is making people get sick. [BO4] Jack destroys the stone. Jack dies. Jack saves the town. [BO5] Claude becomes healthy. [DO6] Pearce and Alain tracks the forest. [DO7] Lupin becomes a werewolf during the night. [DO8] Lupin attacks Pearce and Alain. Alain shots Lupin. Lupin dies. [DO9] Pearce and Alain become heroes in the town.

Story 10 combines plots based on 'Destructive Outsider", "Faust" and "Creation of life" plot templates using the Chinese Boxes weaving strategy:

[DO1] Augustine is a innkeeper. Claude is the major of the town. [DO2] Lupin is an foreigner that arrives to the town. Claude tells Lupin to lodge in Augustine's inn. [CL1] Hubert is a magician. Hubert wants to create humans from animals. [CL2] Hubert captures a wolf and transforms it into a human. Hubert names the new creature as Lupin. [CL3] Hubert teaches Lupin how to behave like a human. [CL4] Lupin becomes a werewolf during the night. Lupin escapes from Hubert's house. [F1] Hubert is frustrated. [F2] Krull is a dark magician. Krull offers Hubert a chance to become powerful. [F3] Hubert accepts Krull's offer. Krull shows Hubert dark magic. [F4] Hubert revives corpses. Hubert has an army of undead. [F5] Hubert realizes that dark magic is bad. [F6] Hubert fights Krull. Hubert turns Krull into stone. [CL5] Hubert looks for Lupin across the country. [CL6] Hubert discovers Lupin near a farm. Lupin kills Hubert. [DO3] Lupin is very kind with Augustine. [DO4] Lupin becomes a werewolf during the night. Lupin kills Augustine. [DO5] Lupin tells Claude that a wolf attacked Augustine. [DO6] Alain asks Pearce for help. [DO7] Lupin becomes a werewolf during the night. [DO8] Lupin attacks Pearce and Alain. Alain shots Lupin. Lupin dies. [DO9] Pearce and Alain become heroes in the town.

B Plot Template Examples

This section presents all the examples of plot templates used throughout the paper. The basis for these templates are the archetypical cinematographic plots catalogued by [1].

The outlines for the plot templates are listed below (more detailed descriptions of the plot templates in terms of scenes and roles are provided in Table 13):

- *The Destructive Outsider*: An outsider arrives to a peaceful community. Although the outsider seems to be a benevolent character, later she/he shows her/his true evil nature. Finally, a hero raises against her/him and saves the community.
- *The Benevolent Outsider*: An outsider arrives to a community in crisis. Although the community considers initially the outsider as a stranger, she/he proves to be a benevolent character by helping the members of the community. Later, she/he sacrifices herself/himself and saves the community.
- *Faust*: Well-known due to Goethe's book [35]. The erudite Faust is highly successful yet frustrated with his life. This situation leads him to make a pact with the Mephistopheles, exchanging his soul for unlimited knowledge and worldly pleasures. Finally, Faust becomes aware of his error and sacrifices himself for achieving the redemption.
- *Descent into Underworld*: The origins of this theme trace the Greek mythology. Orpheus, the legendary musician and poet, traveled to the underworld to desperately safe his dead wife, Eurydice, and bring her back to life.
- *Creation of Artificial Life*: The best-known example of the "Creation of artificial life" plot is Mary Shelley's Frankenstein [31]. Victor Frankenstein, a brilliant young scientist, creates a thinking yet deficient creature in an abominable scientific experiment. The creature escapes from its creator's control and carries out a revenge against him. In the final confrontation, one of the two dies, staying the survivor alone and freed from hate.

C Pseudocode for the Algorithms of the Weaving Strategies

Table 14 lists the pseudo-code for the algorithms corresponding to the plot weaving strategies described in Sect. 4:

- Random Weaving (Algorithm 1)
- Alternating juxtaposed Weaving (Algorithm 2)
- Communicating Vessels Weaving (Algorithm 3)
- Chinese Boxes Weaving (Algorithm 4)

Table 14 Pseudocode for the algorithms of the proposed weaving strategies

```
Data: ps, the list of plots to merge
Result: pm, the merged plot
initialize pm as empty plot;
while ps is not empty do
    randomly select a plot p in ps;
    take the first non-processed scene s in p;
    append s to pm;
    mark s as processed;
    if s is the last scene in p then
        remove p from ps
    end
end
```

Algorithm 1: Random plot weaving algorithm

```
Data: ps, the list of plots to merge
Result: pm, the merged plot
initialize pm as empty plot;
let i the first index in ps;
while ps is not empty do
    let plot p the ith element in ps;
    let scene s the first non-processed scene in
      p;
    append s to pm;
    mark s as processed;
    if s is the last scene in p then
        remove p from ps
    end
    if ps is not empty then
        increase i mod ps size
    end
end
```

Algorithm 2: Juxtaposition plot weaving algorithm

```
Data: ps, the list of plots to merge
Result: pm, the merged plot
initialize pm as empty plot;
randomly select a plot p in ps;
take the first scene s in p;
append s to pm;
mark s as processed;
let i the first index in ps;
while ps is not empty do
    let boolean match FALSE ;
    repeat
        let plot p the ith element in ps;
        let scene s the first non-processed
          scene in p;
        assign to match the result of checking
          the compatibility of s.preconditions
          and sm.postconditions;
        if match then
            append s to pm;
            mark s as processed;
            if s is the last scene in p then
                remove p from ps ;
            end
            assign to i a random index inside
              ps size;
        else
            increase i mod ps size ;
        end
    until match;
end
```

Algorithm 3: Communicating Vessels plot weaving algorithm

```
Data: pp, the primary plot;
ps, the list pf plots to merge
Result: pm, the merged plot
initialize pm as empty plot;
let s the first scene in pp;
repeat
    append s to pm;
    mark s as processed;
    let boolean inserted FALSE;
    foreach plot p in ps do
        if p fits between s and pp next scene then
            add all scenes in p to pm;
            remove p from ps;
            assign to s the last scene in pm;
            assign to inserted TRUE;
            break;
        end
    end
    if not inserted then
        assign to s the next scene in pm;
    end
until ps is empty;
```

Algorithm 4: Frame Story / Chinese Boxes plot weaving algorithm

References

1. Balló, J., Pérez, X.: La semilla inmortal: los argumentos universales en el cine. Editorial anagrama, S.A. Pedro de la Creu, 58, 08034, Barcelona, Spain (2010)
2. Bringsjord, S., Ferrucci, D.: Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine. Psychology Press, New York, NY (1999)
3. Cervantes, M.: Don Quixote. Translated by John Rutherford. Penguin Random House, Random House Tower, New York, USA (2011)
4. Cohen, P.R., Feigenbaum, E.A.: The handbook of artificial intelligence, vol. 3. Butterworth-Heinemann, Oxford (2014)
5. Concepción, E., Gervás, P., Méndez, G.: A common model for representing stories in automatic storytelling. In: 6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017 (2017)
6. Concepción, E., Gervás, P., Méndez, G.: A microservice-based architecture for story generation. In: Microservices 2017 (2017)
7. Concepción, E., Gervás, P., Méndez, G.: Afanasyev: A collaborative architectural model for automatic story generation. In: 5th AISB Symposium on Computational Creativity. AISB 2018 (2018)
8. Concepción, E., Gervás, P., Méndez, G.: Ines: A reconstruction of the charade storytelling system using the afanasyev framework. In: Ninth International Conference on Computational Creativity, ICCC 2018. Salamanca, Spain (2018)

9. Concepción, E., Gervás, P., Méndez, G.: Evolving the ines story generation system: from single to multiple plot lines. In: 10th International Conference on Computational Creativity (ICCC 2019). UNC Charlotte, North Carolina, USA (2019)
10. Concepción, E., Gervás, P., Méndez, G., León, C.: Using cnl for knowledge elicitation and exchange across story generation systems. In: International Workshop on Controlled Natural Language, pp. 81–91. Springer (2016)
11. Dehn, N.: Story generation after tale-spin. IJCAI **81**, 16–18 (1981)
12. Fay, M.P.: Driving story generation with learnable character models. Ph.D. thesis, Massachusetts Institute of Technology (2014)
13. Flaubert, G.: Madame Bovary. Translated by Stephen Heath. Cambridge University Press, UK (1992)
14. Gervás, P.: Propp's morphology of the folk tale as a grammar for generation. In: OASIcs-OpenAccess Series in Informatics, vol. 32. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik (2013)
15. Gervás, P.: Composing narrative discourse for stories of many characters: a case study over a chess game. Literary Linguist. Comput. **29**(4), 511–531 (2014)
16. Gervás, P.: Metrics for desired structural features for narrative renderings of game logs. J. Entertain. Comput. **5**(4), 245–250 (2014)
17. Gervás, P.: Storifying observed events: Could i dress this up as a story? In: 5th AISB Symposium on Computational Creativity. AISB, AISB, University of Liverpool, UK (2018)
18. Gervás, P.: Generating a search space of acceptable narrative plots. In: 10th International Conference on Computational Creativity (ICCC 2019). UNC Charlotte, North Carolina, USA (2019)
19. Gervás, P., León, C.: Integrating purpose and revision into a computational model of literary generation. In: Creativity and Universality in Language, pp. 105–121. Springer (2016)
20. Lebowitz, M.: Creating characters in a story-telling universe. Poetics **13**(3), 171–194 (1984)
21. Lewis, D.: Scorekeeping in a language game. J. Philos. Logic **8**(1), 339–359 (1979)
22. Meehan, J.R.: Tale-spin, an interactive program that writes stories. In: In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, pp. 91–98 (1977)
23. Menéndez, R.: Cinco golpes de genio. Alba Editorial, Barcelona (2013)
24. Pemberton, L.: A modular approach to story generation. In: Proceedings of the fourth conference on European chapter of the Association for Computational Linguistics, pp. 217–224. Association for Computational Linguistics (1989)
25. Perez y Perez, R.: Mexica: A computer model of creativity in writing. Ph.D. thesis, The University of Sussex (1999)
26. Porteous, J., Charles, F., Cavazza, M.: Plan-based narrative generation with coordinated subplots. In: European Conference on Artificial Intelligence (ECAI 2016), vol. 285, pp. 846–854. IOS Press (2016)
27. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press, New York, NY, USA (2000)
28. Riedl, M.O., Young, R.M.: Narrative planning: balancing plot and character. J. Artif. Intell. Res. **39**(1), 217–268 (2010)
29. Rumelhart, D.: Notes on a schema for stories. In: Bobrow, D.G., Collins, A. (eds.) Representation and understanding: studies in cognitive science, pp. 211–236. Academic Press Inc, New York (1975)
30. Sharples, M.: How We Write: Writing As Creative Design. Routledge, London (1999)
31. Shelley, M.: Frankenstein. Penguin Classics, Penguin, UK (2003)
32. Tapscott, A., Gomez, J., León, C., Smailovic, J., Znidarsic, M., Gervás, P.: Empirical evidence of the limits of automatic assessment of fictional ideation. In: C3GI ESSLLI (2016)
33. Thorndyke, P.W.: Cognitive structures in comprehension and memory of narrative discourse. Cogn. Psychol. **9**, 77–110 (1977). https://doi.org/10.1016/0010-0285(77)90005-6
34. Vernet, J. (translator): Las mil y una noches (The Arabian Nights). Editorial Planeta, Barcelona, Spain (1990)
35. Goethe, J. W.: Faust (Parts I and II). Penguin Random House, Random House Tower, New York, USA (2015)
36. Winston, P.H.: The genesis story understanding and story telling system a 21st century step toward artificial intelligence. Tech. rep, Center for Brains, Minds and Machines (CBMM) (2016)

5.5. Artículo: Assessing MultiPlot Stories: from Formative Analysis to Computational Metrics

Este artículo presenta el diseño inicial de un conjunto de métricas de calidad creadas para captar algunas de las observaciones realizadas por evaluadores humanos en un proceso de evaluación formativa sobre la calidad de un conjunto de historias creadas como resultado de la combinación de tramas. Estas métricas puntúan una serie de características de las historias que los evaluadores humanos han identificado como que añaden o quitan valor a las historias evaluadas, principalmente el carácter de los personajes (si son bondadosos o malvados) y la cantidad de acción que ocurre en una escena. Este artículo constituye el primer paso para evaluar cuantitativamente la calidad de las historias multitrama, en la línea de las responsabilidades atribuidas al *Evaluador de Borradores* definido en *Afanasyev*.

5.5.1. Cita completa

P. Gervás, Concepción, E., and Méndez, G. (2021) *Assessing MultiPlot Stories: from Formative Analysis to Computational Metrics*, in Intl. Conference on Computational Creativity, pp. 92-96. Ciudad de México, México.

5.5.2. Resumen original

Recent interest in story generators capable of combining more than on plot line into an elaborate story have been handicapped by the lack of either theoretical material or quantitative metrics to ascertain the quality of outputs of such attempts. The present short paper postulates a set of metrics designed to capture some of the insights elaborated during a formative evaluation of an existing attempt at plot weaving.

Assessing MultiPlot Stories: from Formative Analysis to Computational Metrics

Pablo Gervás, Eugenio Concepción, Gonzalo Méndez
Facultad de Informática
Universidad Complutense de Madrid
Madrid, 28040 Spain
{pgervas,econcepc,gmendez}@ucm.es

Abstract

Recent interest in story generators capable of combining more than on plot line into an elaborate story have been handicapped by the lack of either theoretical material or quantitative metrics to ascertain the quality of outputs of such attempts. The present short paper postulates a set of metrics designed to capture some of the insights elaborated during a formative evaluation of an existing attempt at plot weaving.

Introduction

The mechanics of how to combine more than on plot line into a rich story have become a subject of interest in storytelling research in recent times. Solutions have been proposed to address the task (Fay 2014; Porteous, Charles, and Cavazza 2016; Gervás 2014a; 2018). Yet there is a shortage of either theoretical material or quantitative metrics to ascertain the quality of outputs of such attempts. The present short paper postulates a set of metrics designed to capture some of the insights elaborated during a formative evaluation of an existing attempt at plot weaving. The metrics are calibrated against qualitative evaluations by human judges and tested over outcomes of baseline solutions for plotline weaving. The metrics emulate the observations made by human judges in that they consider separate sets of positive and negative features. The metrics are designed to identify features that at some point in the formative evaluation have been deemed by some human judge to either add or detract to the perceived value of a story. The overall judgment on a given story must be extrapolated from the corresponding collection of features.

Related Work

The three topics considered relevant for this short paper are prior solutions for plot line combination, quantitative metrics for stories and formative evaluation of plot weaving.

Plot Line Combination

The systems reviewed here all combine a number of "plot lines" in some form, but each uses a different terminology for referring to them. To facilitate description, we consider an abstract concept of *plot line* as a sequence of plot elements, each describing an event relevant to the structure of the story, with the possible addition of a set of roles that characters play in the event.

Fay (2014) considers a plot weaving algorithm that builds new stories combining plot lines for a set of given character types. The system finds the character models best matching the given types, retrieves narrative threads associated in the corpus with those models, and finds the best combination of those narrative threads into a single story, ensuring that characters' plots are compatible and that the resulting timeline is consistent.

Porteous, Charles, and Cavazza (2016) presents an interactive storytelling system that constructs stories with multiple interleaved plot lines. Their system constructs the stories dynamically using a plan-based approach in response to set of input parameters that drive the number of plot lines to be interleaved and the relative time spent on presentation of each subplot.

The StoryFire (Gervás 2018) system generates stories inspired by the movements of pieces in a chess game. This system combines concepts of narrative thread – sequence of predicates affecting a given piece – and plot line – a linear sequence of abstract labels for plot-relevant events that may describe an interesting story line. In this case, the plot line (usually single) is used to inform an interweaving of narrative threads for different characters.

Computational Metrics for Stories

Existing previous work on quantitative metrics for stories has not addressed multi-plotline stories explicitly. The work in (Gervás 2014b) describes a number of metrics to quantify a set of desired structural features over narrative renderings of game logs, and it focuses on issues such as coverage of the game, and features like redundancy and continuity of the composed discourse. Earlier works focused on metrics for story novelty (Peinado et al. 2010) and related concepts such as similarity between stories (Hervás et al. 2015). In particular, (Hervás et al. 2015) describes a calibration process based on comparing results on the metric against human judgement.

Formative Evaluation of Plot Weaving

The work of (Concepción, Gervás, and Méndez 2020) explores baseline solutions for weaving together a set of plot

Proceedings of the 12th International Conference on Computational Creativity (ICCC '21)
ISBN: 978-989-54160-3-5

templates into stories where scenes from the different templates appear interleaved. A plot template would correspond to the plot line we are considering – these plot templates include additional information on roles played by the characters. Several procedures for combining plot templates are described, some based on existing literary techniques (Communicating Vessels, Chinese Boxes) and some presented as baselines for computational approaches to the task (subplot concatenation, subplot alternation, and random mixing). A formative qualitative evaluation of 10 story examples is included. This evaluation includes qualitative analyses by human judges of the stories in question, where specific features that add or detract to the perceived value of the story are discussed.

Automated Emulation of Human Assessment of Plotline Weaving

The present short paper describes a set of metrics designed to capture in a numerical form the insights that arose from the formative evaluation presented in (Concepción, Gervás, and Méndez 2020). This formative evaluation uncovered insights at two different levels: features perceivable in stories that are considered valuable by human evaluators, and types of knowledge about the story that are being brought into play by human evaluators when making such judgments.

Insights on Desireable Features in Multi-Plotline Stories

The insights that have been considered relevant for the quality of plot line weaving, and susceptible of numerical formulation are described below.

The comments in the formative evaluation made it clear that there are two features of the stories that play an important role in the perception that human judges have of the quality of their weaving: the valence of characters (whether they are good or evil) and the level of activity conveyed by each scene.

Evaluators praised stories where sub-plots have been combined merging villain with villain or hero with hero.

They also praised stories where descriptive scenes from one plot line were interleaved with descriptive scenes from another, and active scenes were interleaved together. This intuitively leads to a story that switches from a more descriptive mode to a more narrative mode at one point, and the subplots that make it up allign in that sense.

Another feature that was considered relevant is the rhythm of alternation between sub-plots when they are interwoven. Evaluators praised stories in which the rhythm of alternation between subplots – how many scenes from each subplot are told together before switching to the other – matches the perceived impression of activity for the story. If scenes are active, and significant events are happening in each sub-plot, switching between sub-plots can happen every few scenes; whereas if scenes are descriptive and nothing much actually happens in each one, more time should be spent on each sub-plot before switching to another.

Two further features were mentioned as positive for some stories: the existence of an overarching plot for the story that starts and ends the story, and the appearance of a complete sub-plot as an insertion within another.

It is important to note that, when asked to assess stories, human judges did not resort to scoring them or ranking them, rather made a set of observations on each story. These observations were either positive (identifying positive features in the story) or negative (identifying negative features in the story). The metrics that we are proposing follow this same pattern.

Knowledge about Stories Relevant to Multi-Plotline Assessment

The analysis of the formative evaluation suggested that valence of the characters and level of activity of scenes are relevant features that need to be made available to a system hoping to assess multi-plotline stories. Therefore the existing set of resources was hand annotated with values for these features. A baseline annotation was carried out over the templates for sub-plots as a first approximation. In this way, the relevant information is tied in to each plot template.

Valence for characters in a given scene was annotated with a value of -1 for characters performing evil actions and 1 for characters performing good actions. A valence value of 0 is assigned by default to all other characters.

Level of activity of scenes was annotated by adding a flag to scenes in a template that involved some relevant action. The rest of the scenes are considered descriptive.

Quantitative Metrics for Multi-Plotline Weaving

The system as it stands can parse stories written from text files in a particular format into a representation in terms of templates built of scenes. It also allows construction of new stories by combining a number of plot lines using the baseline computational strategies described in (Concepción, Gervás, and Méndez 2020). In both cases the representation that is obtained allows for the automated compilation of numerical data for character valence and activity based on the annotations described.

The procedure constructs four different types of vectors of numerical values for each story:

- *vectors of character valences*: for each character, compile the sequence of valence values for the scenes in the story
- *vectors of sub-plot alternation*: for each span of the story corresponding to a different sub-plot, note which template it comes from
- *vectors of alternation rhythm*: for each span of the story corresponding to a different sub-plot, note its length in number of scenes
- *vectors of matching scene activity*: for each of the spans in the alternation rhythm sequence, compile the count of active scenes

Over these vectors, a number of features considered by the human evaluators can be computed automatically. In all cases, the philosophy is to identify features that at some point in the formative evaluation have been deemed by some human judge to either add or detract to the perceived value of a story.

[CL1] West is a scientist. Lily is West's wife. West lives with his family in the countryside. [DO1] West is a wealthy man; Mary loves West; West loves Mary. [CL2] West dreams with creating artificially a boy. [DO2] Benson is the new steward of West Manor. [DO3] Mary is West's sister. Mary falls in love with Benson. [**DO4**] Benson seduces Mary. [CL3] West creates artificially a boy. West names the boy as Nemo. [CL4] West teaches Nemo to behave as an ordinary boy. [**CL5**] Nemo accidentally kills Mary. Nemo discovers that he is not human. Nemo leaves home. [**DO5**] Benson is discovered by Mary while stealing money. [**CL6**] Nemo steals food in a farm. [CL7] West looks for Nemo. [DO6] West fires Benson. Mary is happy again. [**CL8**] West finds Nemo in an abandoned house in the town. West deactivates Nemo. West is sad and sorry.

Table 1: Story 3 combines a Creation of Life (CL) subplot with a Destructive Outsider (DO) subplot. Scene labels from each subplot are shown in [square brackets], in **bold** if negative in valence. Active events underlined.

[DO1] Augustine is a innkeeper. Claude is the major of the town. [DO2] Lupin is an foreigner that arrives to the town. Lupin hires a room in Augustine's inn. [CL1] Hubert is a magician. Hubert wants to create humans from animals. [CL2] Hubert captures a wolf and transforms it into a human. Hubert names the new creature as Lupin. [CL3] Hubert teaches Lupin how to behave like a human. [CL4] Lupin becomes a werewolf during the night. [**CL4**] Lupin goes to the forest. Lupin kills a deer. [CL5] Hubert follows Lupin's trail. [**CL6**] Hubert discovers Lupin near a farm. Lupin kills Hubert. [DO3] Lupin is very kind with Augustine. [**DO4**] Lupin becomes a werewolf during the night. Lupin kills Augustine. [**DO5**] Lupin tells Claude that a wolf attacked Augustine. [BO1] Claude is sick. Jack arrives in town. [BO2] Jack tells Claude that the town is in danger. Claude does not trust Jack. [BO3] Jack reveals that there is a cursed stone in the town. The curse is making people get sick. [BO4] Jack destroys the stone. Jack dies. Jack saves the town. [BO5] Claude becomes healthy. [DO6] Pearce and Alain tracks the forest. [DO7] Lupin becomes a werewolf during the night. [**DO8**] Lupin attacks Pearce and Alain. Alain shoots Lupin. Lupin dies. [DO9] Pearce and Alain become heroes in the town.

Table 2: Story 9 combines a Creation of Life (CL) subplot with a Destructive Outsider (DO) subplot.

The automatic identification of the following features has been implemented:

- overarching plot (vector of sub-plot alternation starts and ends with the same sub-plot)
- inserted sub-plot (sub-plot appears only once in vector of sub-plot alternation)
- spans with regular interweaving rhythm (a given value of alternation rhythm is maintained over a number of transitions between sub-plots)
- rhythm matched to activity (either slow rhythm for spans with low activity, or high rhythm for spans with high activity)

In addition, the values for valence of characters are used to build an overall pattern of alternation between valences is built for a story. This allows the establishment of distinctions between stories that end events with negative valence (tragedies) and stories that end in events with positive valence (comedies, rags to riches stories, overcoming the monster stories…).

Discussion

The proposed metrics are calibrated against the inspiring stories and tested over automatically generated stories.

Calibration over Inspiring Stories

The results for the proposed metrics over the inspiring stories considered in the formative evaluation of (Concepción, Gervás, and Méndez 2020) are presented in Table 3.

The application of the metrics to these stories is intended as a calibration exercise, to test whether the metrics indeed capture the intuitions that inspired them. Observations on story quality are not considered because the formative evaluation used as reference did not explicitly consider them.

The set of stories includes examples of accepted strategies used in literary text (Chinese Boxes inserts a complete subplot as a single span within another, Communicating Vessels interleaves several subplots with different rhythms). These strategies represent instances of complex weaving strategies that are considered valuable. The metrics clearly identify the Chinese Boxes strategy in stories 6, 7, 9 and 10 (by design the results include both overarching plot and inserted plots).

Story 9 has a span of identified rhythm (*rhytSp* = 1) has a similar situation towards its end (two contiguous spans of 4 scenes) and these also happen to include no activity so they are recognised as a slow pace segment (*slow* = 1) of the story, with relatively slow subplot alternation matching scenes low in activity. Story 3 has a similar situation with spans of 3 scenes, but the activity in that case is not regular. Examples of these stories are shown in Tables 1 (Story 3) and 2 (Story 9). The examples have been chosen using the same subplots to allow comparison of the differences in structure between the resulting complete story.

The Communicating Vessels strategy exercises greater freedom in the way it combines subplots, allowing it to choose whether to include an overarching plot (story 2) or not (stories 5 and 8). Because it interweaves subplots more freely, it can result in a higher number of regular rhythm spans (*rhytSp*, see story 8).

The Alternation strategy by design imposes a fixed rhythm of alternation (*rhytSp*) leading to a single span of regular rhythm of the same size as the story (*spSiz*, see Story 1).

The Random strategy has the potential to replicate the freedom of the Communicating Vessels strategy, as shown by the similar values shown by the metrics for *rhytSp* and *spSiz*.

The patterns for valences show a marked tendency towards positive endings (7/10) over negative ones. This is a natural consequence of the nature of the templates considered (only 1/4 ends on a negative valence). Overall there is a marked tendency to start stories on a negative note (the classical solution of starting with a conflict to be resolved). This again is a result of the set of templates used.

Testing over Generated Stories

The results of testing the proposed metrics over a larger set of automatically generated stories are shown in Table 4.

StID	*Strt*	*#pl*	*ovar*	*ins*	*rhytSp*	*spSiz*	*slow*	*fast*	*valences*
6	B	2	✓	✓	0	[]	0	0	[-1, 1, -1, 1]
7	B	3	✓	✓	0	[]	0	0	[-1, 1, -1]
9	B	3	✓	✓	1	[2]	1	0	[-1, 1, -1, 1]
10	B	3	✓	✓	1	[2]	0	0	[-1, 1, -1, 1, -1, 1]
2	V	2	✓	✗	1	[2]	0	0	[-1, 1]
5	V	2	✗	✗	0	[]	0	1	[-1]
8	V	3	✗	✗	1	[3]	0	0	[-1, 1]
1	A	2	✗	✗	1	[12]	0	0	[-1, 1]
3	R	2	✗	✗	2	[2, 2]	0	0	[-1]
4	R	2	✓	✗	1	[3]	0	1	[-1, 1, -1, 1]

Table 3: Results for metrics for inspiring stories in the formative evaluation. Stories are grouped by strategy: *StID* id in (Concepción, Gervás, and Méndez 2020), *Strt* is strategy used – A alternating, R random, V Communicating Vessels, B Chinese Boxes –, *pl* is number plots, *ovar* overarching plot, *ins* inserted plot, *rhytSp* spans with regular rhythm, *spSiz* sizes of regular rhythm spans, *slow* slow pace span, *fast* fast pace span and *valences* valence pattern.

Strt	*#pl*	*ovar*	*ins*	*rhytSp*	*slow*	*fast*	*v-s*	*v+e*
C	3	0	0	12 (2)	12	0	14	11
C	2	0	0	9 (2)	9	0	11	7
A	3	2	0	8 (19)	0	0	5	10
				6 (18)				
				4 (17)				
A	2	5	0	1 (14)	0	0	9	9
				8 (13)				
				6 (12)				
				5 (11)				
R	3	5	1	1 (10)	7	7	10	6
				1 (7)				
				4 (5)				
				6 (3)				
				14 (2)				
R	2	12	3	2 (6)	2	1	11	10
				2 (5)				
				2 (4)				
				6 (3)				
				6 (2)				

Table 4: Results for the metrics over a set of automatically generated stories: *Strt* is strategy used – C concatenation, A alternating, R random – *#pl* is number of plots, *ovar* number of overarching plots, *ins* number of inserted plots, *rhytSp* number of spans with regular rhythm – number of spans (size of the span) – , *slow* number of stories with slow pace spans, *fast* number of stories with fast pace spans, *v-s* number of negative valence story starts and *v+e* number of positive valence story ends. All values over 20 runs for each case.

The stories are generated using the three baseline computational strategies described in (Concepción, Gervás, and Méndez 2020): concatenation, alternation and random. Results are reported as totals over a set of 20 generated stories for each strategy.

The values for the metrics serve to highlight the shortcomings inherent in the baseline weaving strategies. The Concatenation strategy allows neither overarching plots nor inserted plots. Regularities in rhythm arise by serendipity whenever (at least two of) the subplots involved have the same length. Because the spans involved are always long, the pace of switching between subplots is identified as slow. The Alternation strategy allows overarching plots in certain cases (50% of the time when two plots are used, namely when one of the is longer than the other) and does not allow inserted plots. Due to its nature, it generates a single span with a regular rhythm of alternation of the same size as the story – which varies depending on the templates employed. The templates available do not include continuous sequences of active scenes, so identifying patterns of activity at that rhythm is almost impossible. The Random strategy does allow both overarching plots and inserted plots, and it allows the appearance of spans of regular rhythm in different patterns. The results of this strategy sometimes include several spans of interweaving at different rhythms. In this sense, it is the only of the computational strategies tested that can emulate the behaviour of the Communicating Vessels reference strategy. The Random strategy does have shortcomings of its own in that it is altogether blind to any features that it might be introducing.

With respect to valences, the reported outputs are built using a larger set of templates, with higher prevalence of evil acts towards the end (in 4 out of the 7 templates used). This leads to a higher average of evil ends (slightly over 4 out of 10 as opposed to the 3 in 10 of the hand-crafted stories). The set of templates used is chosen at random, which leads to a more even distribution (5 out of 10 in average) of positive vs. negative beginnings. The values for the hand-crafted stories may have been affected by the original decision to rely on a restricted set of similar templates throughout to make it easier to perceive changes in structure resulting from different weaving strategies.

An example of a generated story is shown in Table 5. This story presents a number of the features that are identified by the metrics proposed in this paper. There is a regular rhythm span with three scenes from the Split Personality Comic subplot (SP1 to SP3) followed by three from the Creation of Life (CL3 to CL5) subplot. There is another regular rhythm span with two scenes from the Split Personality Comic subplot (SP4 and SP5) followed by three from the Creation of Life (CL6 and CL7) subplot. There is no overarching plot, because the story starts with a scene from the Creation of Life subplot and ends with a scene from the Split Personality

[CL1] Scott dreams of bringing to life his recently deceased wife Julia. [CL2] Scott uses a spell to bring to life the portrait of Julia. [SP1] Edward is a physicist researching parallel universes. [SP2] Edward accidentally switches with himself in another universe. In the parallel universe he is Hans, married Martha and working in a farm in the countryside. [SP3] Edward does not know how to take care of the animals and causes various messes. Edward falls in love with Martha. [CL3] Scott talks every evening with Julia. Scott tells Julia that Scott loves Julia. [CL4] Scott falls in love with Edward. Edward falls in love with Scott. Scott tells Julia Scott loves Edward. [**CL5**] Edward visits Scott. Julia is jealous of Edward. Julia takes Edward and disappears. [SP4] Martha realises that Edward is not himself. Edward confesses to Martha what has happened. Martha asks for the original Hans to return. [SP5] Edward begins working on a system to replicate his experiment. Edward manages to communicate with Hans in his original universe. Edward and Hans work together to fix the problem. [CL6] Scott searches for Julia across the castle. [**CL7**] Scott discovers Julia near a farm. Scott kills Julia. [SP6] Edward and Hans manage to reopen communication between universes. Edward returns to his universe and leaves to find his own Martha.

Table 5: Generated Story 2 for Random combination of 2 subplots: combines a Creation of Life subplot and a Split Personality Comic subplot.

Comic subplot. This, together with the choice of subplots, implies that the story has a positive start and a positive end.

The story also includes a number of additional features that are not covered by the metrics but which are clearly relevant for the task. The formative analysis of the stories in (Concepción, Gervás, and Méndez 2020) identified the problem of inconsistency between the life spans of characters unified between two subplots – characters that die in the final story as required by one of the subplots but then continue active as required by another. The story in Table 5 presents two instances of similar phenomena: (1) Edward falls in love with Martha (SP3) and then Edward falls in love with Scott (CL4), and (2) Edward kidnapped (CL5) but remains active without having been released (SP3 and SP4). This type of issue needs to be addressed in further work. It will very likely require further enrichment of the resources with information on when characters are restricted in movement or fall in love.

It is also interesting to note, that, given the peculiarities of Split Personality Comic subplot the characters Edward and Hans are both the same person and separate characters. This complicates computation of this type of consistency restrictions.

Intended Application of the Metrics

The metrics reported here are proposed as a first step towards devising a set of informed weaving strategies that aim to produce stories that exhibit the features identified as desirable. This short paper reports on the enrichment of the underlying resources and the development of the metrics and constitutes a preliminary result. Further work will explore weaving strategies that may take advantage of both of the contributions reported here (enriched knowledge resources and computational metrics for desired features) to achieve multi-plotline stories exhibiting the features deemed valuable by the human judges during the formative evaluation used as inspiration.

Solutions for the automated extraction of the knowledge resources from corpora of stories will also be explored.

Conclusions

This short paper reports these preliminary results on the metrics. The proposed metrics built automatically do serve to identify the features in the inspiring set of stories that they were intended to capture. The baseline solutions for plot weaving considered produce unimpressive output scored low by the metrics. Small peaks in the score do seem to match serendipitous good features observable in the output stories.

Ongoing efforts exist to develop plot weaving solutions to optimise these metrics. It is hoped that such plot weaving solutions will lead to significant improvements in the outcomes.

Acknowledgments

This paper has been partially funded by the projects CANTOR: Automated Composition of Personal Narratives as an aid for Occupational Therapy based on Reminescence, Grant. No. PID2019-108927RB-I00 (Spanish Ministry of Science and Innovation) and InVITAR-IA: Infraestructuras para la Visibilización, Integración y Transferencia de Aplicaciones y Resultados de Inteligencia Artificial, UCM Grant. No. FEI-EU-17-23.

References

Concepción, E.; Gervás, P.; and Méndez, G. 2020. Exploring baselines for combining full plots into multiple-plot stories. *New Generation Computing* 1–41.

Fay, M. P. 2014. *Driving story generation with learnable character models*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Gervás, P. 2014a. Composing narrative discourse for stories of many characters: a case study over a chess game. *Literary and Linguistic Computing* 29(4).

Gervás, P. 2014b. Metrics for desired structural features for narrative renderings of game logs. *Journal of Entertainment Computing*.

Gervás, P. 2018. Storifying Observed Events: Could I Dress This Up as a Story? In *5th AISB Symposium on Computational Creativity*. University of Liverpool, UK: AISB.

Hervás, R.; Sánchez-Ruiz, A. A.; Gervás, P.; and León, C. 2015. Calibrating a metric for similarity of stories against human judgement. In *ICCBR (Workshops)*, 136–145.

Peinado, F.; Francisco, V.; Hervás, R.; and Gervás, P. 2010. Assessing the novelty of computer-generated narratives using empirical metrics. *MINDS AND MACHINES* 20(4):588.

Porteous, J.; Charles, F.; and Cavazza, M. 2016. Plan-based narrative generation with coordinated subplots. In *European Conference on Artificial Intelligence (ECAI 2016)*, volume 285, 846–854. IOS Press.

5.6. Artículo: Evolutionary Construction of Stories that Combine Several Plot Lines

El presente artículo explora una solución evolutiva a la tarea de construir una historia que combine más de una trama en un único discurso lineal. Para ello emplea un conjunto de recursos de conocimiento que capturan las principales características que influyen en las decisiones involucradas (basadas en el modelo de representación de Afanasyev), una representación para el tratamiento evolutivo de los discursos con varias líneas argumentales, y un conjunto de funciones de ajuste basadas en métricas relacionadas con la calidad de los discursos resultantes. Estas últimas, en línea con el servicio *Evaluador de borradores* (*Draft Reflector*) de la arquitectura de referencia. La solución propuesta produce poblaciones de historias con discursos elaborados que combinan varias subtramas.

5.6.1. Cita completa

Gervás, P., Concepción, E., Méndez, G. (2022). *Evolutionary Construction of Stories that Combine Several Plot Lines*. In: Martins, T., Rodríguez-Fernández, N., Rebelo, S.M. (eds) Artificial Intelligence in Music, Sound, Art and Design. EvoMUSART 2022. Lecture Notes in Computer Science, vol 13221, pp. 68-83. Springer, Cham.

5.6.2. Resumen original

Although the narrative structure of common entertainment products like Hollywood movies or TV series is generally composed of a number of different plot lines combined into a single narrative discourse, efforts on computational modeling of story generation have to this point focused mostly on the construction of stories with a single plot line. The present paper explores an evolutionary solution to the task of building a story that combines more than one plot line into a single linear discourse. This requires: a set of knowledge resources that capture the main features that influence the decisions involved, a representation suitable for evolutionary treatment for discourses with several plot lines, and a set of fitness functions based on metrics related to the quality of the resulting discourses. The proposed solution produces populations of stories with elaborate discourses that combine several subplots.

Evolutionary Construction of Stories that Combine Several Plot Lines

Pablo Gervás( ), Eugenio Concepción, and Gonzalo Méndez

Facultad de Informática, Universidad Complutense de Madrid, 28040 Madrid, Spain
{pgervas,econcepc,gmendez}@ucm.es
http://nil.fdi.ucm.es

Abstract. Although the narrative structure of common entertainment products like Hollywood movies or TV series is generally composed of a number of different plot lines combined into a single narrative discourse, efforts on computational modeling of story generation have to this point focused mostly on the construction of stories with a single plot line. The present paper explores an evolutionary solution to the task of building a story that combines more than one plot line into a single linear discourse. This requires: a set of knowledge resources that capture the main features that influence the decisions involved, a representation suitable for evolutionary treatment for discourses with several plot lines, and a set of fitness functions based on metrics related to the quality of the resulting discourses. The proposed solution produces populations of stories with elaborate discourses that combine several subplots.

Keywords: Story generation · Multiplot stories · Evolutionary approach

1 Introduction

The narrative structure of common entertainment products like Hollywood movies or TV series is generally composed of a number of different plot lines combined into a single narrative discourse. There are very clear mechanisms at work in putting multiple plot lines together into a successful narrative discourse in this fashion, yet efforts on computational modeling of story generation have to this point focused mostly on the construction of stories with a single plot line. This is in part due to the application of a traditional rule of engineering – do not consider complex versions of the problem until the simple versions have been solved – and in part due to oversight of another such rule – if the artifact you are trying to build is made up of several parts, the construction process should

This paper has been partially funded by the projects CANTOR: Automated Composition of Personal Narratives as an aid for Occupational Therapy based on Reminescence, Grant. No. PID2019-108927RB-I00 (Spanish Ministry of Science and Innovation) and InVITAR-IA: Infraestructuras para la Visibilización, Integración y Transferencia de Aplicaciones y Resultados de Inteligencia Artificial, UCM Grant. No. FEI-EU-17-23.

T. Martins et al. (Eds.): EvoMUSART 2022, LNCS 13221, pp. 68–83, 2022.
https://doi.org/10.1007/978-3-031-03789-4_5

understand what such elementary parts are and how they come together. The present paper explores an evolutionary solution to the task of building a story that combines more than one plot line into a single linear discourse. This requires a set of knowledge resources that capture the main features that influence the decisions involved – namely a set of basic plot lines as sequences of scenes, each annotated with the characters that take part, the narrative roles they play, and basic semantics such as when characters are born, die, or fall in love. A representation suitable for evolutionary treatment is proposed for discourses with several plot lines, and a set of fitness functions are defined based on metrics related to the quality of the resulting discourses are proposed. The solution defined over these elements produces populations of stories with elaborate discourses that combine several subplots.

The combination of several plot lines into a discourse is more complex than a simple weaving. Plot lines are not actually independent streams of narrative discourse that get combined together into a complex fabric: the character sets from different subplots combined into a story do not remain distinct, they may be merged so that the same character in the overall story often plays different roles in more than one of the subplots. This point is addressed explicitly in the solution proposed in this paper. The process we want to model involves an additional operation of merging (some of the characters in) the narrative threads for the different plotlines by instantiating their narrative roles (either main or secondary) with the set of characters for the overall plot.

2 Related Work

Three topics are considered relevant for this paper: prior solutions for plot line combination, quantitative metrics for stories, and evolutionary approaches to creation of narratives of some kind.

2.1 Plot Line Combination

In recent years there has been an increase in the number of research efforts that consider the construction of stories with more than one plot line. In reviewing research efforts on this topic, we have opted for unifying the terminology under an abstract concept of *plot line* considered as a sequence of plot-relevant elements that make sense in the order in which they appear in the story, and linked by at least a shared set of protagonist and secondary characters. We will refer to the plot-relevant elements in a plot line as *scenes*. When more than one plot line are involved in a larger story, each of these plot lines is considered a *subplot* of the story.

Two different approaches have been followed to create multi-plotline stories. In the first one, the story is created by adding scenes incrementally, switching between different plot lines so that each of the plot lines involved is also dynamically constructed. In the second one, a set of existing plot lines is considered

at the start, and the construction process involves deciding how these plot lines will be combined in the discourse for the final story.

Under the first approach, Porteous et al. [16] describe an interactive storytelling system that builds stories in a way that the resulting story will show multiple interleaved plot lines. Their system relies on a plan-based approach that takes input parameters that govern the number of plot lines that should be involved in the final version and how much of the final story should taken up by each plot line.

Under the second approach, systems first compile a set of narrative threads from a given source and then search for possible combinations of them that make a story. Fay [3] constructs multi-plotline stories by combining together narrative threads for specific types of characters found in a given story request, so that main characters in the story request are bound to secondary characters in threads for other characters and the elements from each thread are combined into a consistent overall timeline. The Raconteur [6] and StoryFire [5] systems obtain narrative threads from a conceptual description of a chess-game, and builds stories by selecting a *plot schema* to use as a template to be filled in with matching scenes from the available threads. The PlotAssembler system [7] builds stories by combining a set of small spans of narrative discourse called *axes of interest* in ways that interweave their scenes, ensuring that character continuity across scenes is compatible with probabilities mined from a corpus of prior stories. The work of Concepción et al. [2] explores baseline solutions for weaving together a set of plot lines into stories. The plot lines considered include additional information on roles played by the characters. The set of weaving procedures includes some based on existing literary techniques and some presented as baselines for computational approaches to the task.

2.2 Computational Metrics for Stories

Over the past few years a body of computational metrics for stories has been developed. As the set of features that might be considered in a story is vast, these metrics generally focus on particular aspects that are relevant for specific purposes. When the goal is to invent a new story, metrics relevant to the purpose focus on aspects that may be related to originality: story novelty [14], similarity between stories [9] and correlations between easy-to-measure features in the stories and the creativity attributed to them by human judges [20].

More recent work addresses the evaluation of multi-plotline stories specifically. An attempt at formative qualitative evaluation of multi-plotline stories was carried out in [2]. A group of human judges were asked to consider the quality of 10 examples of multi-plotline stories, and to identify specific features that they considered positive or negative contributions to the perceived value of the story. A number of interesting conclusions were drawn. First, the most detrimental feature to perceived story quality was the existence of semantic inconsistencies in the story, such as characters that keep taking part in the story after being dead. Second, positive judgments often involved identification of characteristics of the story that were optional rather than necessary, such as the existence of an

overarching plot, or the fact that one subplot was inserted fully within another. In [8], the qualitative analyses presented as part of the formative evaluation were distilled into a set of metrics designed to capture them in a numerical form. These metrics were designed to respond to the set of features that human judges were seen to focus on when asked to assess multi-plotline stories, and to translate the judgments made onto numerical scores.

2.3 Evolutionary Construction of Narratives

Without pretending to be exhaustive, this section reviews some of the existing efforts to generate narratives by means of evolutionary solutions in terms of two different aspects: which elements they combine and what type of fitness function they employ.

In terms of the elements that are combined to make stories, McIntyre and Lapata [13] present a plot generator that builds a plot from the set of entities appearing in a sentence provided by the user. For each entity in the sentence the system retrieves a plot line from a knowledge based automatically extracted previously, and creates a space of possible stories involving the given entities by merging together these plot lines. This search space is then explored by means of genetic algorithms, using as fitness function a combination of story coherence and story interest. Gómez de Silva and Pérez y Pérez [19] propose a model of story construction that combines the MEXICA existing knowledge-based story generator [15] with the GENCAD evolutionary approach for the adaptation stage in case-based solutions to architectural problems [18]. The story construction model relies on the knowledge-based heuristics of MEXICA to build initial populations and applies the evolutionary approach to refine them. Among the features that the fitness functions are designed to detect are individuals with 'incestuous ancestry', built by applying more than once rules from the same exemplar, which may lead to repetition of events in the output story.

In terms of the fitness functions employed to drive the evolutionary approach, Fredericks and DeVries [4] describe an application of an evolutionary solution driven by novelty search [11] to improve the novelty of procedurally-generated small narratives fragments for text-based games. Kartal et al. [10] use Monte Carlo tree search to drive planning-based narrative generation in support of a interactive framework for user-driven narrative variation. They rely for selection on a combination of the believability of the resulting story and the percentage of the user-defined goals the current story accomplishes. Soares de Lima et al. [12] combine planning with an evolutionary search strategy guided by story arcs to generate quests for games. Candidate quests are constructed by a planner as linear sequences of events or tasks to be accomplished by the player. The evolutionary algorithm chooses among them using a fitness function that builds for each candidate quest the sequence of tensions that arise from it, and scores it based on its match with a target curve of evolving tensions provided by the user.

3 An Evolutionary Multiplot Story Composer

A story built by combining plot lines requires a complex set of decisions: in which order to combine the scenes from the different plot lines and how to merge the characters sets from the different plot lines into a single set of characters for the overall story. The search space of possible solutions is immense. Prior attempts to explore this search space in terms of heuristics for informing the required decisions [2] have shown that stories produced in this way are significantly more rigid in structure than human produced stories. An evolutionary solution based on fitness functions that capture some of the desired features for the output stories may provide efficient means of traversing this search space.

3.1 The Knowledge Resources

The experiments reported in this paper are carried out over a set of plot lines extracted from a compilation of classic plots used in cinema [1]. A subset of these plots have been distilled into a set of *plot templates*, which are formalised knowledge resources that describe the structure of the plot as a sequence of scenes. The set of plot templates currently available is: The Benevolent Outsider, The Creation of Artificial Life, Descent into the Underworld, The Destructive Outsider, Faust, Split Person Comic, Split Person Tragic.

Each *scene* describes the characters that take part, the narrative roles they play, the set of semantic annotations that are used by the system to check consistency, and a template for rendering the scene as text. The narrative roles played by the characters are represented by labels that identify the main characters involved in a given plot – say, Creator and Creature in a Creation of Artificial Life plot, or Tempter and Tempted in a Faust plot. The semantic annotations cover aspects about the meaning of the scene that may affect the perceived consistency of the stories in which they appear. The current version considers the following semantic annotations: whether characters are *created/born or die* in the scene and whether characters *fall in love or out of love*. These semantic annotations allow the definition of metrics to identify situations where narrative roles from different subplots are assigned to the same character in the story, and events affecting the different roles are incompatible – characters dying more than once, or serially falling in love. These metrics are used to inform the fitness function for the evolutionary solution.

Each plot template has a particular character identified as the *protagonist* – this is usually one of the characters playing an important role in the plot.

This set of features emerges from the analysis of a number of multi-plot stories described in [2] and the metrics for automated assessment of quality of multi-plotline stories presented in [8]. The insights from these analyses have identified these features as relevant to the perception of quality of a multi-plotline story.

3.2 Character Fusion and Discourse Planning

The analyses carried out to this point on how multi-plotline stories are put together show that, given a set of subplots to be considered as inputs, there are two different processes that contribute to the final result.

The first process is related to the set of characters that take part in each subplot. For a story to be successful, it appears to be important that the sets of characters involved in each subplot have a non-empty intersection. In operational terms, this is achieved by unifying some character a from subplot A with character b from another subplot B, so that a single character – say *John* – in the story undertakes both the part of character a from subplot A and the part of character b from subplot B. We refer to this operation as *character fusion*. In terms of traditional views on natural language generation [17] this operation is a part of the task of *content determination*, since the nature of subplots is somewhat changed in the process of attributing particular characters to the variables in the corresponding template.

The second process is related to the order in which the scenes from the different subplots are presented in the discourse for the final story. This is known to be crucial to the impact that the story has on the audience, and exploited intentionally by authors to achieve effects such as *suspense* – where presentation of certain information is withheld on purpose – or *cliff-hanger breaks* – where transitions to cover other subplots are made at points where tension is high in the current subplot. In terms of traditional views on natural language generation [17] this operation is a fundamental part of the task of *discourse planning*.

Character fusion and discourse planning are tightly interconnected by the information in the semantic annotations for scenes. A fusion between a character a from one template A and a character b from another template B is only possible if it will not result in a final discourse in which the fused character is seen to behave in an impossible manner – such as being born or dying more than once, or being involved in more than one passionate romance without the fact being addressed by the story.[1] The metrics that inform the fitness function should penalise the scores of stories that incur in this type of inconsistency. Even if none of these extreme inconsistencies occur, a particular discourse plan may be incoherent if a fused character takes part in events that happen in the final discourse either before its birth or after its death. Specific metrics for identifying these situations are also needed.

3.3 Representing Multiplot Stories for Evolutionary Construction

In order to apply an evolutionary approach, each story draft needs to be represented in terms of some kind of genetic information that describes how it addresses the main tasks involved: discourse planning and character fusion. To achieve this we propose a solution in several parts.

[1] For clarifications on how romantic conflicts are handled in the current version of the system see the discussion in Sect. 3.6.

The problem of constructing a multiplot line story considers the following inputs: a set of plot lines to combine, where each plot line is defined by a narrative thread expressed as a sequence of scenes together with a set of characters that appear in it.

This implies that, for a particular problem of combining N plotlines, the length of the final discourse is determined by the total number of scenes in the narrative threads being considered, and the maximum number of possible characters featuring in the story is determined by the union of the sets of characters in the narrative threads being considered.

For simplicity, we are assuming that the relative chronological order of the scenes in each narrative thread is respected in the final discourse. This is not a necessary condition. Indeed, many stories present instances of altered chronology (flashbacks, flashforwards). However, we leave this point to be addressed in further work.

The characterisation of the discourse plan for a given story candidate requires: establishing which narrative thread to start on, defining the specific points in the final sequence in which the discourse switches to a different narrative thread, and defining to which of the other available threads the discourse is switching when it does.

The representation is based on the fact that, for a given story construction problem, the length of the final discourse is known and fixed, and the fact that the set of plotlines being considered is known and fixed.

The information on discourse planning is represented in terms of vectors that define how the narrative threads for the different plot lines are combined into the final linear discourse:

- a single digit (0 or 1) defines which narrative thread the final discourse starts with
- a sequence of digits (0 or 1) defines for the total number of scenes in the final discourse whether the next scene follows on with the prior narrative thread or it switches to a different thread
- a sequence of digits (ranging between 1 and N − 1, where N is the total number of plots being combined) defines how many of the available plots are skipped whenever the discourse switches to a different narrative thread

The set of possible characters for the complete story is defined by the union of the sets of variable names for the characters appearing in each narrative thread. These variable names need to be distinct across the different narrative threads to avoid confusion. This is ensured by assigning a prefix with the plotline name to all the variable names that feature in any given narrative thread.

The caracterisation of the choices for character fusion for a given story candidate requires an assignment of character names to each of the variables in the joint set of variables for the story.

For simplicity, the set of potential character names for the story is defined to be the set of integers from 0 to C, with C being the cardinality of the joint set of variables for the story. This is sufficient to represent any choices made in

terms of character fusion (with variables in two different positions in the name-assignment vector being assigned to the same integer). The form of the resulting stories would be significantly improved by a later stage of transforming these integer names for the characters into strings representing realistic names.

An example of representation is shown in Fig. 1.

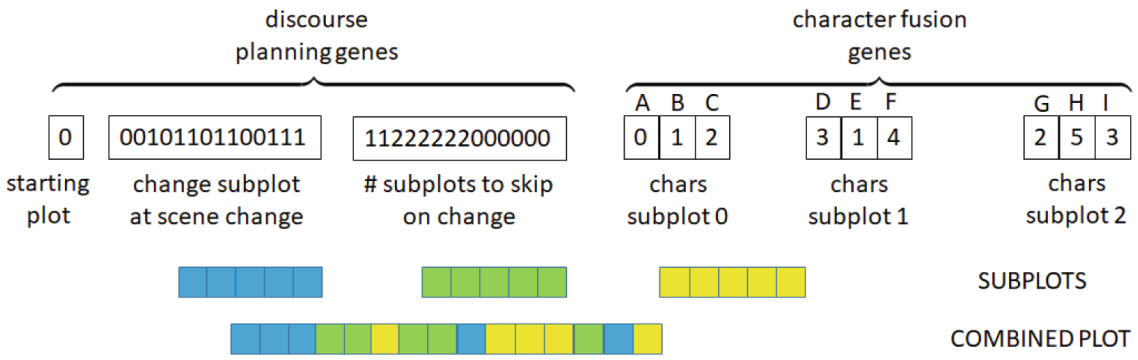


Fig. 1. Genetic representation for a combination of three plots of length 5, each with 3 characters. Fuses characters B (p0)/E (p1), C (p0)/G (p2) and D (p1)/I (p2).

3.4 Constructing an Initial Population

An initial population of story candidates is built by assigning values to the representation described in Sect. 3.3. For each of the different parts of the representation the process of assignment of values needs to be treated differently.

For the initial digit that defines which narrative thread to start on, and for the vector of decisions on whether to switch, random choice between 0 and 1 is suitable.

For the vector of decisions on skip size at each switch, random choice between 1 and $N-1$ (with N the total number of plots being combined) is suitable.

For the vector of decisions of which character to assign to each variable, the choice is more complex. This is because variables from the same narrative thread should not be assigned to the same character, at the risk of confusing the relations between characters in the corresponding subplot. The process of assignment is carried out separately for the set of variables for each thread. For such a set of variables, the process decides at random whether to assign to each variable either a character name chosen at random from those already used in some of the narrative threads already processed, or an entirely new character name chosen at random from the character names that remain free. This ensures the required constraints are satisfied.

3.5 Evolutionary Operators

Once a population has been constructed, mutation and cross over operators are applied to it.

Because of the different nature of the various parts of the representation, specific operators of each kind are applied to the different parts.

For the mutation operators:

- for the starting point gene, the value is mutated at random
- for the switch point vector, values at each point are either mutated or not depending on a threshold parameter
- for the skip size vector, values at each point are either mutated or not depending on a threshold parameter, and, if required, mutated to a value chosen at random within the required range
- for the character assignment vector, character names at each point are either mutated or not depending on a threshold parameter, and, if required, mutated to a character name chosen at random within the required range

For the cross over operators:

- for the starting point gene, the value of the two individuals being considered is swapped
- for the switch point vector, a point in the vector is chosen at random and the corresponding halves of the vectors for the two individuals are swapped over
- for the skip size vector, a point in the vector is chosen at random and the corresponding halves of the vectors for the two individuals are swapped over
- for the character assignment vector, the assignments of characters for the two different individuals are swapped over

3.6 Fitness Functions

The construction of potential story drafts as described in the initialisation of the population and the evolutionary operators is mostly random. However not all possible combinations are actually valid. Certain plot lines, when combined in certain orders, give rise to incoherence in the semantics of the story. Incoherence may affect different aspects: characters that act before being born or after being dead, or characters that fall in love serially with no regard for previous romances. It falls to the fitness function to differentiate between valid and invalid story candidates. This is achieved by taking into account the knowledge resources described in Sect. 3.1.

As there are many different aspects to be considered for a story, a modular approach is applied to the definition of a fitness function for the system. Rather than build a complex fitness function that addresses all the aspects, a number of targeted fitness functions are built, and the fitness for a given individual is computed as a function of the results it achieves under the specific fitness functions.

This approach allows a differentiated consideration of the aspects related to validity of the story draft, the aspects related to satisfaction of optional traits, and the aspects that may actually relate to perceived quality of any given solution.

The different aspects that need to be considered as contributing to the quality of stories differ in the way they impact the overall perception of a reader. These different ways require different solutions to model numerically the expected behaviour. The solution presented here is a tentative proposal that captures the basic intuitions arising from prior empirical studies. Further work may need more detailed empirical studies and matching adaptations of the computational solutions.

Validity Fitness Functions. A first set of fitness functions addresses the consistency of the final discourse in terms of the semantics for the scenes as annotated in the knowledge resources described in Sect. 3.1. All these functions assign a score of 100 if the story is consistent with respect to the corresponding feature, and 0 otherwise.

The current set of fitness functions includes the following aspects that affect the validity of a story draft:

- whether characters are born more than once in the story (rules out combinations of plot lines that merge into a character two subplots that both mention the birth)
- whether characters die more than once in the story (rules out combinations of plot lines that merge into a character two subplots that both mention the death)
- whether characters are active in the story at points that do not lie between their birth and their death (or the boundaries of the story if birth or death are not mentioned)
- whether the romantic behaviour of characters is consistent (rules out combinations of subplots that imply one character is passionately in love with different people)

The fitness function for consistent romantic behaviour is currently an initial approximation to the task. Cases where characters fall in love with more than one person do exist as valid stories, but they are only interesting when the plot addresses the romantic conflict explicitly and resolves it in some way. The limited procedure of story construction applied here – restricted to interweaving scenes from two different plots – cannot address this task in its current form. Further work will consider extensions to the construction procedure and a matching revised fitness function to address this issue.

Optional Traits Fitness Functions. The second set of fitness functions addresses aspects noted to be positive traits by the human judges but which constitute optional rather than necessary configurations for a story. These traits add value to a story when present but do not detract from its quality if absent.

Fitness functions are defined to capture the corresponding traits, also scoring 100 if the trait is present and 0 if it is absent.

In the present version of the system, such fitness functions are added to the mix for a given run only if the corresponding trait is desired in the outcome population.

The following traits are considered for discourse plans:

- *inserted subplot*: a complete subplot is inserted as an aside into the story (all scenes from the subplot appear together in the final discourse, and surrounded by scenes from other subplots)
- *overarching plot*: the story starts and finishes with scenes from the same subplot, which therefore appears as a frame for any other subplots in the story – this does occur in cases where there is an inserted subplot but may also occur in other situations, and therefore it is considered as a separate feature

Additional traits may be considered in the same way for character fusions, which establish a link between the casts of the subplots involved in any given story. The importance of the link depends on the relative importance of the character involved in the link with respect to each of the subplots. For the present paper, the following type of links have been considered:

- *shared protagonist*: the same character acts as protagonist of two subplots
- *stitching protagonist*: the protagonist from one subplot plays an important role – different from the corresponding protagonist – in another

Fitness functions of the type explained above are defined for each of these types of links, allowing the outcomes to be driven towards stories satisfying the corresponding criteria.

Combining Fitness Functions to Score Individual Stories. To ensure that the different types of fitness functions described above are combined into a single score for each candidate story in a population, stories are assigned as a final score the average of the scores for the set of fitness functions for specific features selected.

This ensures that individual stories that are invalid or do not exhibit the desired traits disappear from the population as early as possible, and that stories with higher quality have a higher chance of survival.

4 Discussion

The results of the proposed system are presented and the relation of the proposed approach with previous work is discussed.

4.1 Results

The proposed system is run in each case with an initial population of 1,000 individuals generated at random, with the described operators for mutation (probability of mutation set to 0.2) and crossover (probability of cross over set to 0.05), for 20 generations. At each generation populations are culled by selecting the next generation using a best scoring criterion.

Table 1. An example of output story combining the plotlines creation of artificial life (CLM) and split person tragic (SPT). Columns show: the *text* for the story broken down by scenes, description of the *discourse plan* indicating for each scene which plotline it comes from, description of the cast indicating the roles they play in the respective plot lines and what names have been assigned to them – which reflect any *character fusions* present. Columns 1 and 2 are aligned by scenes, column 3 applies to the story as a whole.

Text	#	Plotline	Scene name	Variable	Role	Name
A is a doctor researching life improving serum. A has dedicated his whole life to science and has no friends or family	0	SPT	Introduction	CLM-0	Frankenstein	C
A accidentally creates a serum that changes him into B, an evil twin of himself	1	SPT	Splitting	CLM-1	Creature	D
C is a German nobleman. C lives in a big castle. C is sad. C 's wife died recently. C loves his dead wife. C dreams of bringing to life his wife	2	CLM	Conception of idea	CLM-2	-	E
C studies book of magic. C discovers a spell to bring to life a portrait of his wife. C brings to life D, the portrait of his wife	3	CLM	Creation of new life	SPT-0	DrJekyll	A
C talks every evening with D. C tells D that C loves D. D is kind to C	4	CLM	The being complies with its master's will	SPT-1	MrHyde	B
C goes to a ball. C meets E in a ball. C falls in love with E. E falls in love with C. C tells D C loves E	5	CLM	Rebellion and escape	SPT-2	-	C
E visits C. D is jealous of E. D takes E and disappears	6	CLM	Infringement by creature			
C searches for D across the castle	7	CLM	Quest for the creature			
D tells C that C must defeat D to save E. C burns the portrait. D is destroyed and E appears magically. C and E are happy	8	CLM	Death of the creature			
B brutally kills some friends of A	9	SPT	Entanglements			
C realises that B is an evil version of A. A confesses to C what has happened	10	SPT	Discovery			
A decides to stop taking the serum	11	SPT	Unravelling			
A transforms involuntary into B. B goes on a rampage. Having lost control, A decides to commit suicide, killing B in the process	12	SPT	Denouement			

Some examples of results are shown below.

The stories generated by the system are rendered as text using the templates stored for the corresponding scenes in the knowledge resources, replacing the variables with the assigned character names. An example of complete story is shown in Table 1. Numbers used internally as character names have been replaced with capital letters for ease of reading. This story is obtained with the configuration set to construct stories with an inserted plot. The discourse plan indeed shows all scenes from CLM plot appearing together, bracketted by scenes from SPT. In this case, the two subplots are only very lightly connected by character fusions: variable CLM-0, the protagonist of CLM, playing the role of Frankenstein has been fused with variable SPT-2 that corresponds to a secondary character in SPT.

To illustrate the operation of the fitness functions for optional traits, examples of the structure of different stories are shown in Table 2.

Table 2. Structures for stories obtained with different configurations for the optional traits concerning discourse structure, using different combinations of plotlines creation of artificial life (CLM), split person tragic (SPT), Faust (FA) and descent into the underworld (DiU). Column 1 shows a story with inserted plots, column 2 shows a story with overarching plot but no inserted plot, column 3 shows a story with neither overarching plot nor inserted plot. For ease of understanding of the structure of the discourse plans, the scenes of one of the subplots are highlighted in bold.

#	Plotline	Scene name	#	Plotline	Scene name	#	Plotline	Scene name
0	SPT	Introduction	0	SPT	Introduction	0	DiU	Lovers Initial Happiness
1	SPT	Splitting	1	**FA**	Frustrated character regrets his life	1	DiU	Lost
2	**CLM**	Conception of Idea	2	SPT	Splitting	2	DiU	Mourn and quest for the beloved
3	**CLM**	Creation of new life	3	**FA**	Temptation	3	**FA**	Frustrated character regrets his life
4	**CLM**	The being complies with its master's will	4	**FA**	Pact with evil	4	**FA**	Temptation
5	**CLM**	Rebellion and escape	5	SPT	Entanglements	5	**FA**	Pact with evil
6	**CLM**	Infringement by creature	6	**FA**	Evil actions	6	DiU	Deal and brief reunion
7	**CLM**	Quest for the creature	7	**FA**	Enlightenment	7	**FA**	Evil actions
8	**CLM**	Death of the creature	8	SPT	Discovery	8	DiU	Infringement of compromise
9	SPT	Entanglements	9	**FA**	Redemption	9	DiU	Metamorphosis
10	SPT	Discovery	10	SPT	Unravelling	10	**FA**	Enlightenment
11	SPT	Unravelling	11	SPT	Denouement	11	**FA**	Redemption
12	SPT	Denouement						

Table 3. Examples to illustrate options on character fusion. Names of fused characters are highlighted in bold.

DUW-0	Orpheus	**A**	**A**	A
DUW-1	Eurydice	B	**B**	B
DUW-2	Hades	D	C	**C**
FAU-3	Faust	**A**	**B**	**C**
FAU-4	Mephistopheles	B	**A**	D

To illustrate the operation of the fitness functions for character fusion, examples of different stories are shown in Table 3. To provide informative views while respecting page limit constraints, only cast descriptions are shown. The first column is an example of shared protagonist (same character A plays Orpheus and Faust). The second column is an example of stitching protagonist (character A plays Orpheus and Mephistopheles and character B plays Euridice and Faust). The third column is a weaker example of stitching protagonist (character C plays Hades and Faust).

4.2 Relation with Previous Work

The character fusion operation considered here is comparable to binding between characters as used by Fay [3].

The procedure applied in McIntyre and Lapata [13] shows significant parallels with our own approach: a set of possible stories is created by combining plot lines from different stories and then an evolutionary approach is applied to search for a convincing set of output stories. However, our system differs in that it takes as input a set of already established plot lines that cannot be altered, whereas McIntyre and Lapata explore different choices of elements – tree structures corresponding to sentences – to build the stories.

The metric for detecting individuals with 'incestuous ancestry' in [19], intended as it is to avoid repetition of events in the stories, is related to our fitness function to avoid characters in the stories falling in love more than once. Attempts to combine more than two plotlines may require a similar adaptation to avoid, at least initially, stories that include the same plot line several times.

The semantic annotations for the plot templates in our knowledge resources play a similar role to the domain database as considered in [12] to compute tension. The judgement on validity of stories based on these semantic annotations plays a role similar to the believability metric as considered in [10].

5 Conclusions

The evolutionary approach to constructing multi-plotline stories provides efficient means of building a population of drafts that satisfy constraints on semantic validity over the final linear discourse for the story. The drafts in the population

can be steered towards stories that satisfy specific features in terms of particular structures in the discourse plan (such as overaching plot or inserted plots) and/or particular choices for character fusion (such as shared protagonists, linking roles or only loose connections between the casts of the different subplots).

The current set of features identified as relevant to story quality is not well suited to numerical ranking in terms of fitness functions. As a result, final populations tend to converge towards scores of 100/100 for all individuals. Nevertheless, the fact that scores for individuals are constructed as a combination of more specific scores for particular features implies that during the construction procedure – in earlier generations of the evolution – individuals do have scores in the range between 0 and 100. This allows the evolutionary procedure to explore different combinations of the various subparts of the representation vector, to achieve a set of valid solutions in the final population that satisfy the desired constraints.

As future work we will consider extending our system with metrics of the various types used in the other approaches reviewed: believability, story arcs based on tension.

Although user-defined goals are a feature specific to user-driven narratives of the type addressed in [4] and [10], stories built outside these contexts often do have a purpose beyond entertainment – convincing, educating,... Moreover, there may not be a single goal but rather a set of goals to achieve. Under this light, extending the fitness functions for our system with metrics for percentage of goals achieved will be considered as future work.

It seems that evolutionary approaches have been used often for story generation as a secondary process applied to a set of stories constructed previously by some other method. Examples of this initial material produced by other means are the stories built by MEXICA in [19] and the grammar-driven quest sketches produced by Tracery in [4]. In our case, we apply the evolutionary solution to combine plot templates that constitute already-built stories. The solution we propose could be used in future work as a secondary process on the results of preceding story generation procedures of a different type.

References

1. Balló, J., Pérez, X.: La semilla inmortal: los argumentos universales en el cine. Ed. Anagrama, Barcelona (2007)
2. Concepción, E., Gervás, P., Méndez, G.: Exploring baselines for combining full plots into multiple-plot stories. New Generation Computing **38**(4), 593–633 (2020). https://doi.org/10.1007/s00354-020-00115-x
3. Fay, M.P.: Driving story generation with learnable character models. Ph.D. thesis, Massachusetts Institute of Technology (2014)
4. Fredericks, E.M., DeVries, B.: (Genetically) improving novelty in procedural story generation. arXiv preprint arXiv:2103.06935 (2021)
5. Gervás, P.: Storifying observed events: could i dress this up as a story? In: 5th AISB Symposium on Computational Creativity. AISB, AISB, University of Liverpool, UK, April 2018

6. Gervás, P.: Composing narrative discourse for stories of many characters: a case study over a chess game. Literary Linguist. Comput. **29**(4), 511–531 (2014)
7. Gervás, P.: Generating a search space of acceptable narrative plots. In: 10th International Conference on Computational Creativity (ICCC 2019). UNC Charlotte, North Carolina, USA, July 2019
8. Gervás, P., Concepción, E., Méndez, G.: Assessing multiplot stories: from formative analysis to computational metrics. In: 12th International Conference on Computational Creativity (ICCC 2019), Mexico City, Mexico, September 2021
9. Hervás, R., Sánchez-Ruiz, A.A., Gervás, P., León, C.: Calibrating a metric for similarity of stories against human judgement. In: ICCBR (Workshops), pp. 136–145 (2015)
10. Kartal, B., Koenig, J., Guy, S.J.: User-driven narrative variation in large story domains using Monte Carlo tree search. In: Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems, AAMAS 2014, pp. 69–76. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC (2014)
11. Lehman, J., Stanley, K.O.: Exploiting open-endedness to solve problems through the search for novelty. In: Proceedings of the Eleventh International Conference on Artificial Life. MIT Press (2004)
12. de Lima, E.S., Feijó, B., Furtado, A.L.: Procedural generation of quests for games using genetic algorithms and automated planning. In: 18th Brazilian Symposium on Computer Games and Digital Entertainment, SBGames 2019, Rio de Janeiro, Brazil, 28–31 October 2019, pp. 144–153. IEEE (2019). https://doi.org/10.1109/SBGames.2019.00028
13. McIntyre, N., Lapata, M.: Plot induction and evolutionary search for story generation. In: Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pp. 1562–1572. Association for Computational Linguistics, Uppsala, Sweden, July 2010. https://aclanthology.org/P10-1158
14. Peinado, F., Francisco, V., Hervás, R., Gervás, P.: Assessing the novelty of computer-generated narratives using empirical metrics. Minds Mach. **20**(4), 588 (2010). https://doi.org/10.1007/s11023-010-9209-8
15. Pérez y Pérez, R., Sharples, M.: Mexica: a computer model of a cognitive account of creative writing. J. Exp. Theor. Artif. Intell. **13**(2), 119–139 (2001)
16. Porteous, J., Charles, F., Cavazza, M.: Plan-based narrative generation with coordinated subplots. In: European Conference on Artificial Intelligence (ECAI 2016), vol. 285, pp. 846–854. IOS Press (2016)
17. Reiter, E., Dale, R.: Building Natural Language Generation Systems. Cambridge University Press, Cambridge (2000)
18. Gómez de Silva Garza, A.: An evolutionary approach to design case adaptation. Ph.D. thesis, The University of Sydney, Australia (2000)
19. de Silva Garza, A.G., y Pérez, R.P.: Towards evolutionary story generation. In: Colton, S., Ventura, D., Lavrac, N., Cook, M. (eds.) Proceedings of the Fifth International Conference on Computational Creativity, ICCC 2014, Ljubljana, Slovenia, 10–13 June 2014, pp. 332–335. computationalcreativity.net (2014). http://computationalcreativity.net/iccc2014/wp-content/uploads/2014/06/15.3_Garza.pdf
20. Tapscott, A., Gomez, J., León, C., Smailovic, J., Znidarsic, M., Gervás, P.: Empirical evidence of the limits of automatic assessment of fictional ideation. In: C3GI ESSLLI (2016)

Capítulo 6

Discusión y aportaciones

"Ask courageous questions. Do not be satisfied with superficial answers"
— Carl Sagan

6.1. Contexto de la discusión

La presente sección recoge una discusión relativa a las decisiones adoptadas y las aportaciones del trabajo investigador a partir del desarrollo de la tesis. Los principales aspectos revisados están vinculados a la consecución de los objetivos de la presente investigación a partir de los resultados presentados a lo largo de los capítulos previos.

En la primera sección se realiza un análisis de los sistemas basados en la arquitectura de referencia y el modelo de representación del conocimiento propuestos por Afanasyev. Estos sistemas se focalizan en objetivos distintos, pero a lo largo de la comparativa se muestra cómo la flexibilidad y la capacidad de integrar planteamientos distintos del marco proporcionado por Afanasyev permite la construcción de sistemas generadores consistentes. En este análisis se destaca además qué aspectos de la arquitectura de referencia y del modelo de representación son aplicados, y cuáles no, en cada caso. En la siguiente sección se realiza una comparación del uso realizado por cada sistema de la arquitectura de referencia y del modelo de representación de conocimiento, y se termina el capítulo con una serie de consideraciones acerca de la consecución de los objetivos tecnológicos planteados al inicio a la luz de la comparación realizada entre los sistemas desarrollados.

6.2. Análisis de los sistemas desarrollados en términos de la arquitectura y la representación de conocimiento propuestas

En la presente sección se comparan la arquitectura de referencia y el modelo de representación del conocimiento definidos en Afanasyev, descritos en los capítulos

3 y 4, con las implementaciones planteadas en tres sistemas diferentes. Los tres elementos a comparar serán una evolución simplificada de Charade (Méndez et al., 2016), que denominaremos "Charade++" por distinguirla del sistema original; el sistema INES, que es una evolución completa del mismo Charade (Concepción et al., 2018b); y finalmente la evolución de INES para generar historias con múltiples tramas (Concepción et al., 2019), que denominaremos INES 2.

Las figuras 6.2 y 6.1 representan, respectivamente, el modelo de representación del conocimiento y la arquitectura de referencia de Afanasyev. Se presentan como marco de referencia para que resulte más cómoda la identificación de los diversos usos de los mismos en los tres planteamientos descritos a continuación.

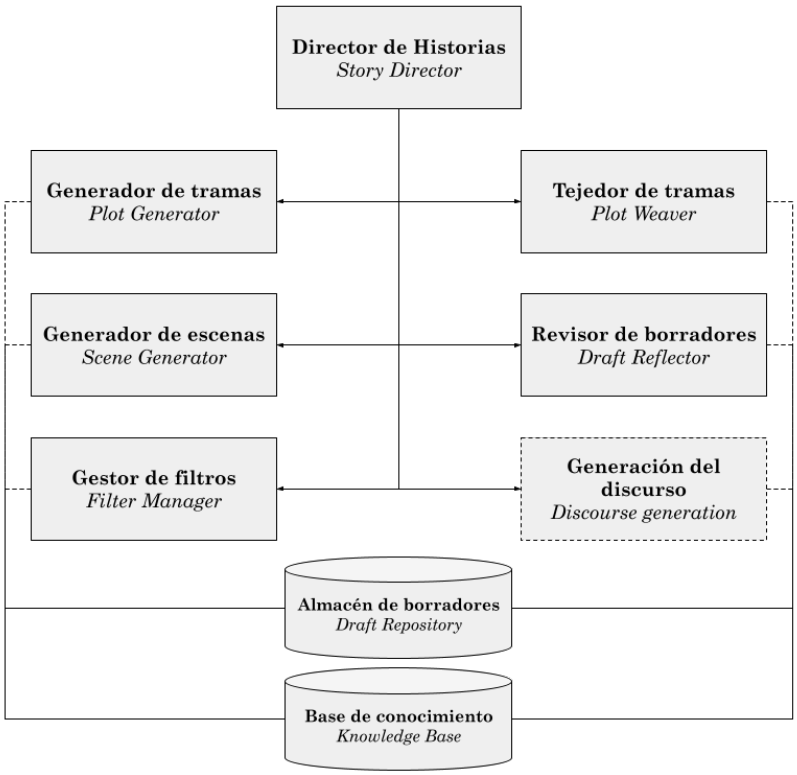


Figura 6.1: Arquitectura de referencia de Afanasyev

6.2.1. Análisis de Charade++

El sistema presentado en esta sección fue una primera aproximación a la evolución de Charade hacia un sistema más complejo mediante la aplicación del modelo Afanasyev. Se trata de un modelo interesante para comparar la aplicación flexible tanto del modelo de representación del conocimiento como de la arquitectura de referencia de Afanasyev.

Charade (Méndez et al., 2016) originalmente era un sistema centrado en simular las interacciones y emociones de personajes que interactuaban de forma continua.

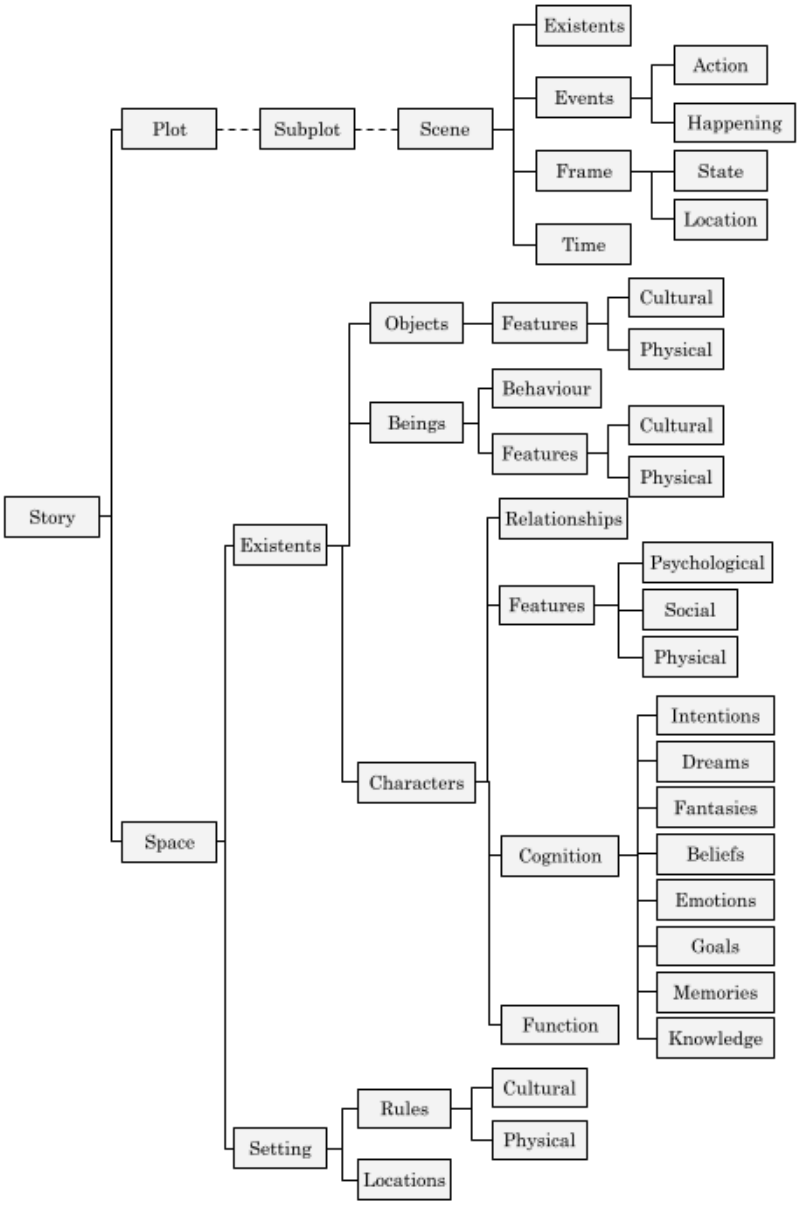


Figura 6.2: Modelo de representación del conocimiento de Afanasyev

Las historias generadas eran una secuencia de interacciones entre personajes y cambios en sus sentimientos hacia los otros. En este sentido, estas historias no tienen estructura con trama y escenas. Como se puede ver en la figura 6.4, los aspectos de la representación del conocimiento de Afanasyev necesarios para trasladar desde Charade hasta Charade++ se centran en los personajes, sus relaciones y su emotividad. Para incorporar la estructura de trama y escenas a Charade++ se plantea generar una trama inicial y cada interacción del conjunto de personajes es una escena. Así, las

acciones realizadas por los personajes durante cada interacción entre ellos constituye el contenido de una escena.

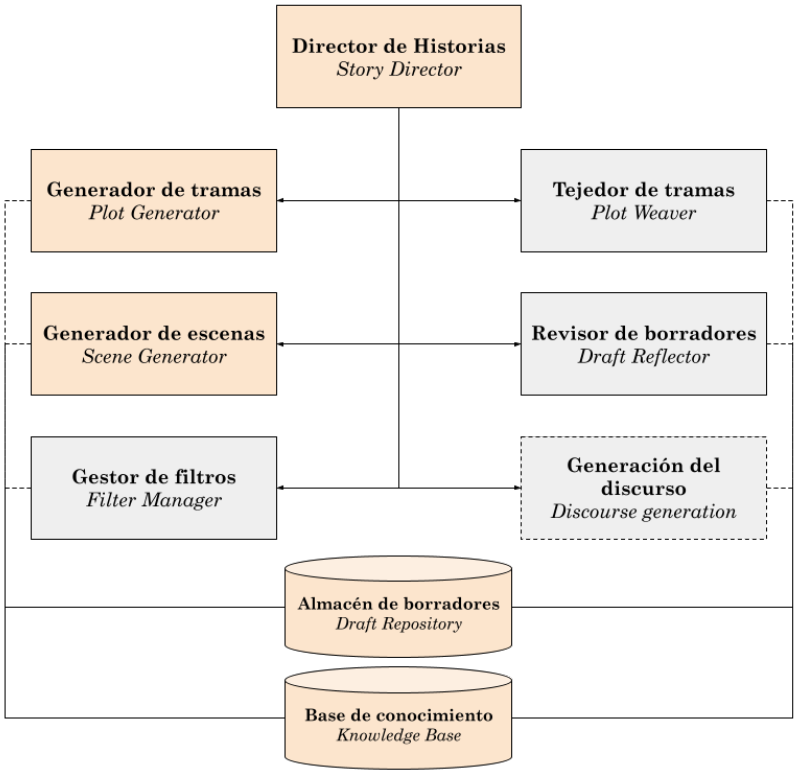


Figura 6.3: Arquitectura de referencia de Charade

En la figura 6.3 se muestran los componentes de la arquitectura de referencia de Afanasyev que intervienen en Charade++. A partir de las indicaciones referidas en el párrafo anterior, cada ciclo de interacción entre los personajes constituye una escena; por tanto, la lógica original de simulación de Charade está contenida en el generador de escenas (*Scene Generator*).

La operación completa del conjunto fue diseñada como sigue:

1. El director de historias invoca al generador de tramas para que cree una trama.

2. El generador de tramas construye una trama base, con un número aleatorio de escenas (vacías de contenido inicialmente), y unas precondiciones y poscondiciones para cada escena generadas de forma aleatoria. Esto queda reflejado en un primer borrador que sigue la estructura de representación del conocimiento vista en la figura 6.4.

3. El generador de escenas es invocado por el director de historias tantas veces como escenas hay en la trama. Cada vez que se invoca al servicio, el generador de tramas ejecuta una simulación con los personajes existentes en la representación del conocimiento. Para seleccionar qué acciones puede ejecutar cada

personaje, toma en consideración las restricciones en forma de precondición y postcondición que se aplican sobre el elemento estado de la representación del conocimiento. Así, si hay una indicación para que los personajes ejecuten acciones "amistosas", en la simulación se tomarán en consideración como acciones permitidas todas aquellas que estén etiquetadas como tales.

4. El director de historias verifica que todas las escenas han sido completadas y da el borrador como completado y lo marca como historia.

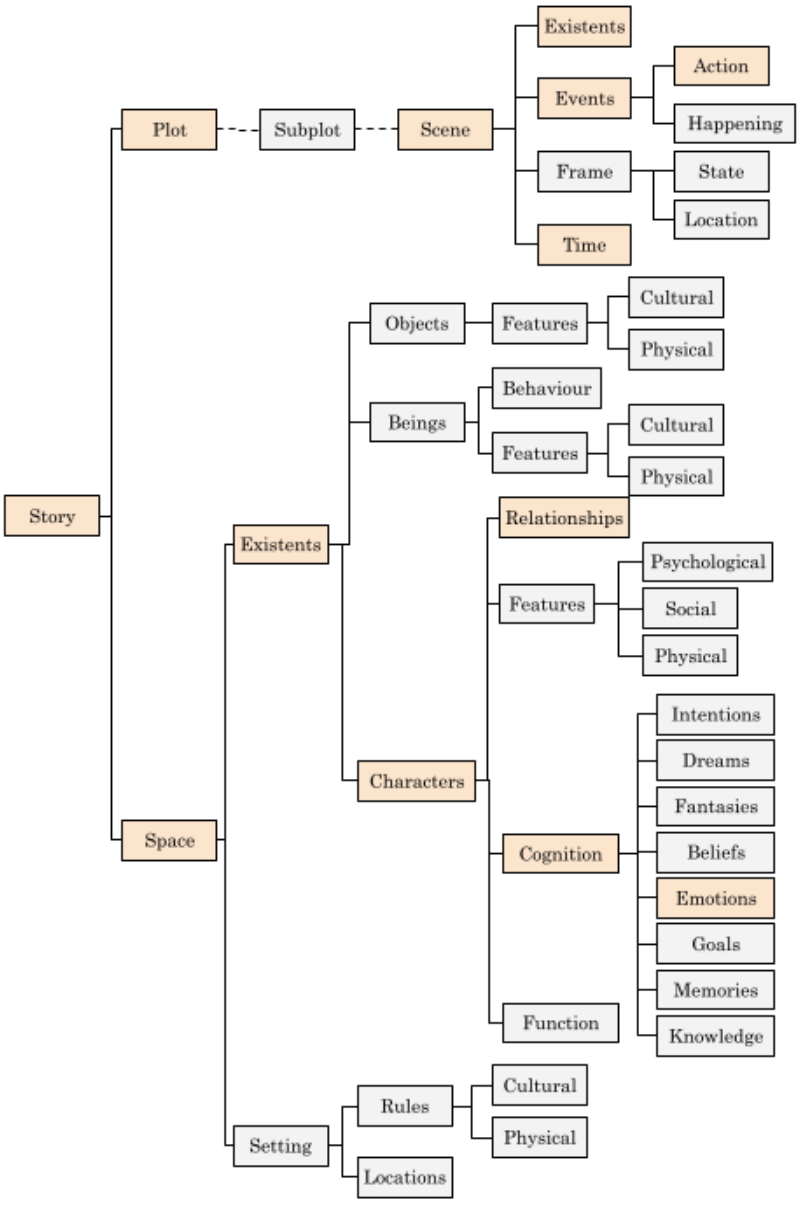


Figura 6.4: Modelo de representación del conocimiento empleado en Charade

Como se puede ver, el comportamiento propuesto para el sistema es básico. No se aprovechan elementos complejos de la representación, como la parte de reglas físicas o las ubicaciones en el mundo. La secuencia se basa únicamente en una línea temporal, no causal. Y los diferentes episodios no tienen por qué guardar una consistencia entre sí, dado que las condiciones pre y post que definen lo que puede suceder o no, se generan de forma aleatoria. Tampoco se consideran eventos que no tengan que ver con las acciones realizadas por los personajes. Y la caracterización de estos últimos es tremendamente básica. Se puede concluir que el aprovechamiento del total de elementos de la representación del conocimiento es pequeño. Y sin embargo, esto no resulta un problema, dado que el diseño de la representación es flexible y soporta esta parcialidad. Por otro lado, el aprovechamiento de la arquitectura es igualmente pequeño. El generador de tramas se implementa como un mero generador aleatorio. La mayor parte de la lógica significativa en este caso se encuentra en el generador de escenas, donde la implementación se basa en el motor de Charade adaptado para que sea capaz de interpretar y aplicar las restricciones al comportamiento de los personajes fijadas en la representación de las escenas (pre y postcondiciones).

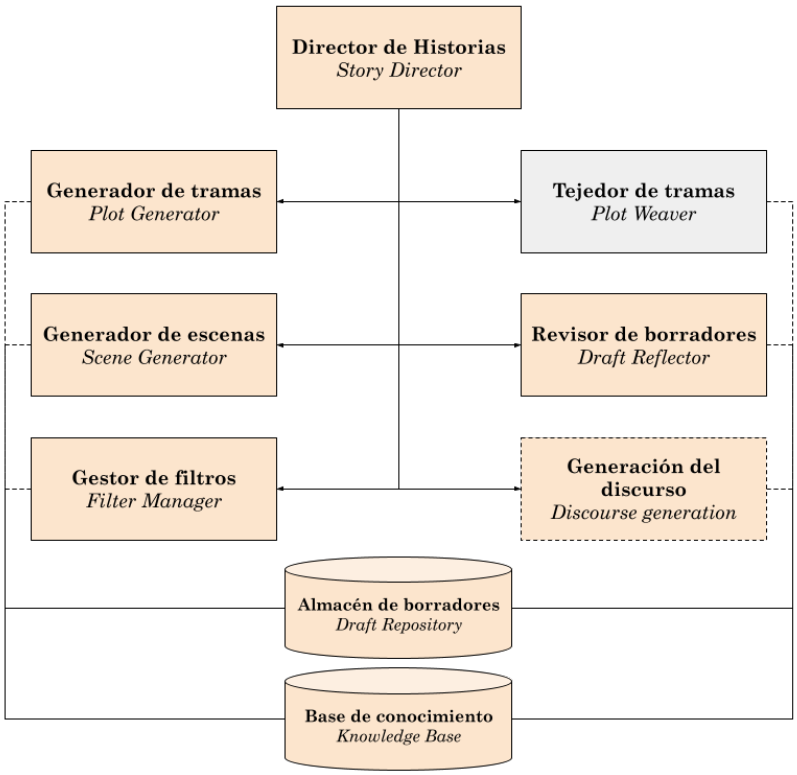


Figura 6.5: Arquitectura de referencia de INES

6.2.2. Análisis de INES

INES es un sistema desarrollado a partir de la arquitectura de referencia de Afanasyev. A diferencia de Charade++, es un sistema harto más complejo, con un planteamiento más ambicioso, especialmente en el aspecto relativo a la generación de la trama y a su ligazón con los episodios que la componen. No se realizará en esta sección una descripción detallada de INES, dado que ya se ha presentado en el Capítulo 4. El objetivo de esta sección es destacar aquellos aspectos de INES que ponen en valor la arquitectura de referencia y el modelo de representación de Afanasyev, así como compararlo con los otros dos sistemas referidos en la sección.

Como se puede ver en la figura 6.6, se hace un aprovechamiento más exhaustivo de los recursos proporcionados por la representación. El hecho de que la arquitectura emplee prácticamente todo el catálogo de servicios definidos en la arquitectura de referencia (ver figura 6.5) hace que se pongan en juego dichos recursos. Por ejemplo, la mayor riqueza en la generación de la trama por parte del generador de tramas pone en juego todos los elementos relativos al contexto de cada escena en la representación del conocimiento (estado, localización, tiempo, acciones y acontecimientos). Por otro lado, se mantiene y se amplía (incluyendo información sobre creencias) el marco de Charade++ relativo a las relaciones y las emociones de los personajes. Y por el lado del espacio de la historia, se toman en consideración restricciones relativas a las reglas del mundo de la historia.

Template "The Destructive Outsider"

Role	**Description**
Outsider	This character is a newcomer to the community. Initially, she/he performs friendly actions, but later her/his evil nature is revealed
Community member	Any of the characters playing this role behaves friendly with the others until the beginning of the outsider's evil actions, at which point the lack of trust becomes widespread
Heroe	Any of the characters playing this role is also a community member. They behaviour differ from the rest of community members when they raise to face and defeat the evil outsider

Plot Structure

Scene	**Description**
Peaceful community	Initial state introducing the members of a peaceful community
Arrival	The arrival of the outsider to the community
Outsider destructive actions	The outsider acts against the members of the community, performing destructive actions, without being uncovered
The outsider revealed	The true evil nature of the outsider is revealed
The rise of the heroes	The The heroes rise from the community and fight against the outsider
Purge	The outsider is purged. The community becomes peaceful again

Tabla 6.1: "El intruso destructor" ("The Destructive Outsider"), ejemplo de plantilla tomada de Concepción et al. (2020)

El generador de tramas definido en Afanasyev, **Plot Generator**, ha sido implementado en INES como un microservicio etiquetado como "Audrey". Este componente toma su nombre de la actriz Audrey Hepburn, que interpretó el papel principal en "Charade" (Donen, 1963). Tal como se describe en (Concepción et al., 2018b), es un generador de tramas basado en plantillas que produce esquemas a partir de un subconjunto de tramas básicas cinematográficas recopiladas por Balló y Pérez (2007). El procedimiento de construcción de la trama comienza seleccionando una de las

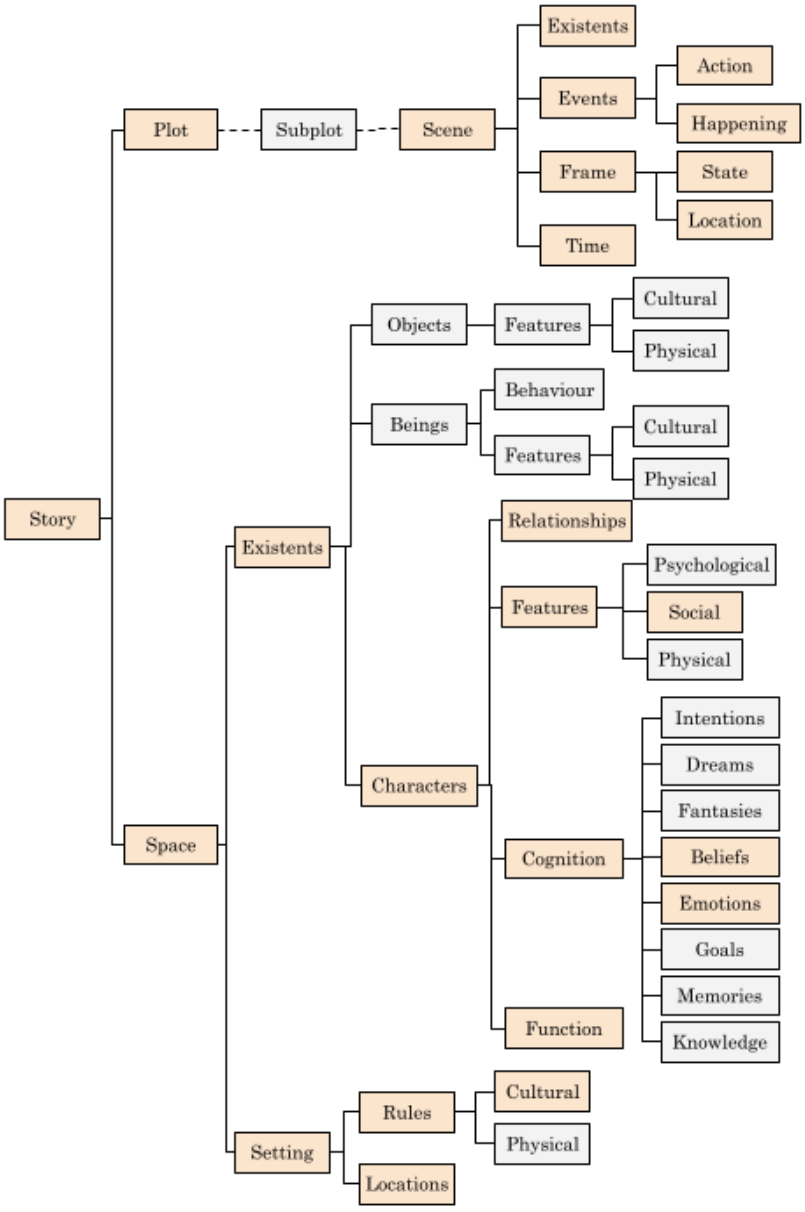


Figura 6.6: Modelo de representación del conocimiento empleado en INES

plantillas predefinidas, que consisten en una estructura conceptual con la forma de la trama. La plantilla puede seleccionarse al azar o puede elegirse según el nombre de la plantilla recibido como parámetro. Una plantilla de trama es una descripción abstracta de una unidad de material argumental internamente coherente que representa una posible subtrama y que puede utilizarse como bloque de construcción para tramas más elaboradas. Básicamente consiste en una disposición de escenas genéricas en las que una lista de personajes participan en ciertos roles, relevantes para esa

subtrama en particular, realizando ciertos tipos de acciones. Los roles y los tipos de acciones que deben realizar constituyen un conjunto de restricciones que la trama generada debe cumplir. La tabla 6.1 muestra un ejemplo de plantilla de trama que ha llevado a generar una historia como la que se resume en la tabla 6.2.

Story

Title	Beware of the fiancee

Story space (Setting + Characters)

Setting	North America mid 20th Century Family
Character	**Role**
Anne, the new fiancee of Scott	Outsider
Edward, a wealthy businessman and head of the family	Community Member
Sarah, the elder daughter of Edward	Community Member and Hero
Scott, the youngest son of Edward	Community Member
Samuel, a friend of Scott	Community Member

Plot (based on template "The Destructive Outsider)"

Scene	**Actions**
Scene 1: A peaceful community	Scott and Samuel work together on Edward's factory. Edward is proud of Scott. Sarah secretly loves Samuel. Edward, Sarah and Scott live together in a villa. Samuel visits Scott.
Scene 2: The arrival of the outsider	Scott introduces Anne to Sarah and Edward. Scott gives a ring to Anne. Edward welcomes Anne. Sarah welcomes Anne. Edward invites Anne to dinner.
Scene 3: Outsider destructive actions	Anne steals a jewel from Edward's room. Anne tells Edward that Samuel stole a jewel.
Scene 4: Conflict	Edward accuses Samuel of stealing the jewel Sarah defends Samuel. Samuel gets angry with Scott.
Scene 5: The outsider revealed	Sarah witnesses Anne trying to steal a swatch . Sarah tells Edward that Anne is a thief. Edward tells Scott that Anne is a thief. Scott gets angry with Edward.
Scene 6: The rise of the heroes	Sarah faces Anne. Scott demands Anne to leave. Anne leaves the town.
Scene 7: Conclusion	Scott says sorry to Edward. Edward says sorry to Samuel. Samuel gives thanks to Sarah. Samuel falls in love with Sarah. Samuel ask Sarah to marry him.

Tabla 6.2: Ejemplo de historia con una trama tomada de Concepción et al. (2020).

6.2.3. Análisis de INES 2

INES 2 es la evolución de INES para generar historias con múltiples tramas, tal como se describe en los artículos *Evolving the INES story generation system: from single to multiple plot lines* Concepción et al. (2019) y *Exploring Baselines for Combining Full Plots into Multiple-plot Stories* (Concepción et al., 2020), recogido en el Capítulo 5.

Como puede verse en la figura 6.8, la arquitectura del sistema se focaliza en la generación de tramas y su entretejido, prescindiendo de las etapas de generación de escenas. Igualmente, este interés se detecta también en la forma en que se representa el conocimiento, vista en la figura 6.7.

El objetivo principal de la implementación de INES 2 era validar las técnicas de generación multitrama descritas en el Capítulo 5 y poner a prueba para ello

la flexibilidad de la arquitectura de referencia y el modelo de representación del conocimiento de Afanasyev.

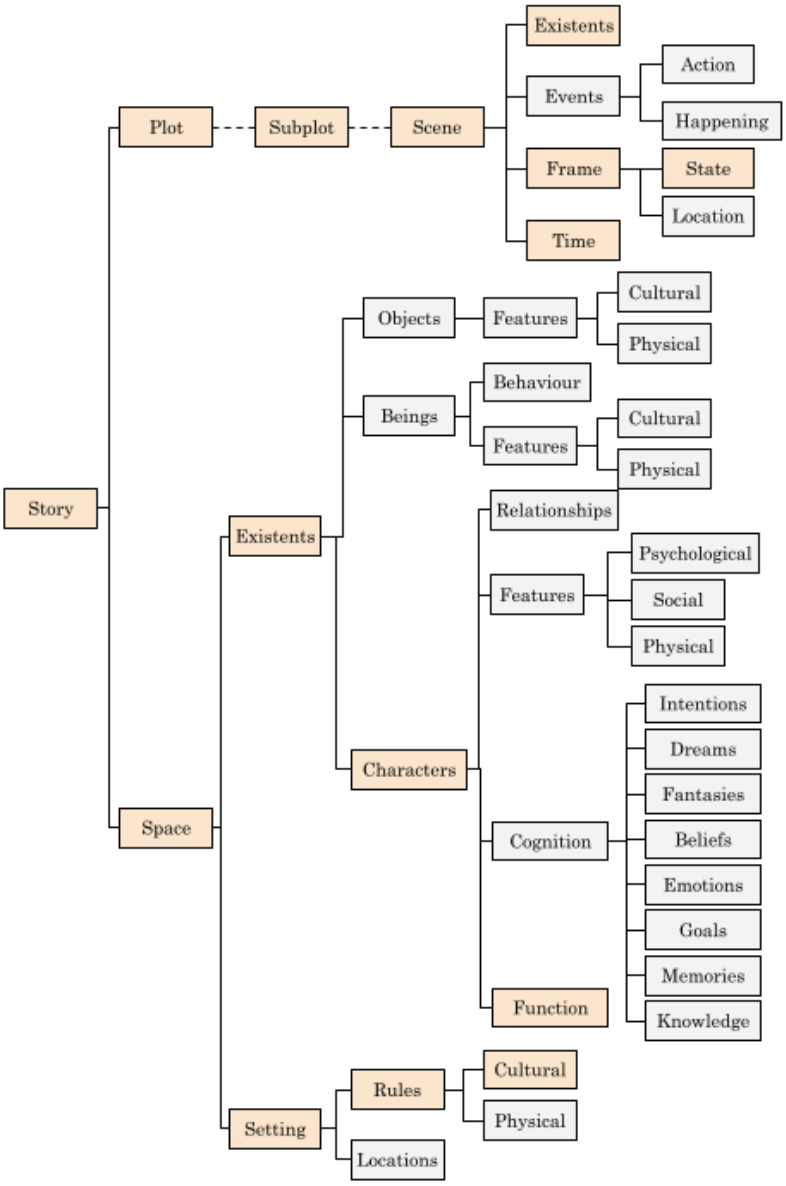


Figura 6.7: Modelo de representación del conocimiento empleado en INES 2

De acuerdo con dicho objetivo, y dado que la efectividad del modelo de generación de escenas y los aspectos de la representación del conocimiento correspondientes se habían probado en INES, el foco en INES 2 se centra en la generación de múltiples tramas para una misma historia, su adecuada representación como subtramas, y la generación de una salida donde las diferentes subtramas se entretejen de forma consistente para crear una línea argumental.

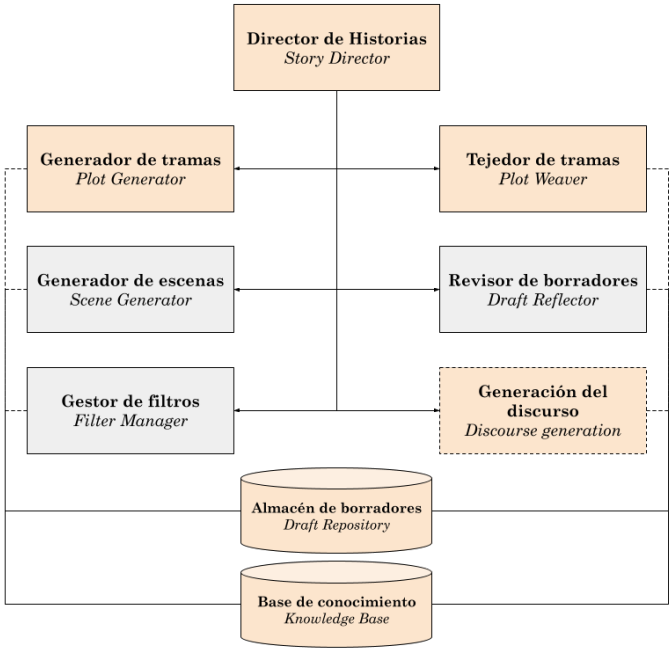


Figura 6.8: Arquitectura de referencia de INES 2

A nivel arquitectónico, el componente clave es la implementación del tejedor de tramas. Este servicio es el encargado de entretejer las diferentes subtramas para crear una secuencia de escenas procedentes de las mismas que tenga consistencia al mismo tiempo que preserva la sensación de estar leyendo varios hilos dentro de la misma historia.

Las técnicas de entretejido implementadas por dicho componente están descritas con detalle en el Capítulo 5, por lo que no se revisarán nuevamente. A nivel de operación del componente, es importante repasar cómo funciona dentro del conjunto:

1. Para generar una historia con múltiples tramas, el director de historias realiza varias peticiones al generador de tramas. Tantas como tramas se deseen incluir en la historia.

2. El generador de tramas va creando nuevas subtramas en la representación de la historia a medida que recibe peticiones. El proceso de generación de las tramas se basa en el mecanismo de plantillas empleado por INES. Es importante destacar que hay ciertas condiciones que se deben cumplir: todas las subtramas deben compartir el mismo espacio narrativo (el nodo *Space* en el modelo de representación de la figura 6.7), y los conjuntos de personajes de cada una de las subtramas no pueden ser disjuntos (deben compartir al menos un personaje).

3. Una vez que el director de historias detecta que se han generado todas las tramas (subtramas) a entretejer, realiza una petición al tejedor de tramas, indicándole la heurística que debe aplicar (Aleatoria, Alternante, Cajas chinas o Vasos comunicantes).

4. El tejedor de tramas combina las subtramas aplicando la técnica deseada y genera una historia con una trama combinada donde se suceden las escenas conforme al resultado de entretejer las escenas de todas las subtramas de entrada.

5. El director de historias realiza una petición al servicio de generación de lenguaje natural para volcar a texto la historia.

Como se puede ver, a diferencia de INES, donde se completaba el proceso de generación con el detalle de las escenas, en el caso de INES 2 la generación llega sólo hasta el entretejido de tramas, puesto que son los objetivos fijados para el sistema. No obstante, debido a la compatibilidad entre los componentes de los dos sistemas, merced a su ajuste a la arquitectura de referencia de Afanasyev, es posible conectar el generador de escenas de INES a la arquitectura de INES 2, de forma que se genere el correspondiente detalle de las escenas. Esto último puede verse en el ejemplo recogido en la tabla 6.3, donde se combinan dos tramas y se completan con acciones las escenas correspondientes.

Para completar los ejemplos, que pueden consultarse en detalle en (Concepción et al., 2020), se incluye otro ejemplo de generación de historia con tres tramas en la tabla 6.4.

Destructive Outsider	**Faust**	**Actions**
Peaceful community		Mary and John work together on their farm. William helps John with the farm tasks. Mary invites William to dinner. Jeff visits John. Jeff gives a present to John.
	Frustrated character regrets his life	Jeff feels miserable. Jeff thinks that he is weak. Jeff hates Carlson. Jeff wants to arrest Carlson.
Arrival of the outsider		Adam arrives at the city. Adam buys a ranch. Jeff welcomes Adam. John welcomes Adam. Mary invites Adam to dinner.
	Temptation	Adam offers help to Jeff. Adam offers money to Jeff. Adam tells Jeff to arrest all the gunmen.
Outsider destructive actions		Adam wants John's farm. Adam sneakily burns down John's barn.
	Pact with evil	Adam blames Carlson for burning down John's barn. Jeff accepts Adams'money. Jeff arrests Carlson.
Conflict		Adam offers Mary to buy her farm. Mary accepts Adam's offer. John refuses to sell his farm. John gets angry with Mary.
	Evil actions	Adam shoots John. John is injured.
Outsider revealed		Mary witnesses Adam shooting John . Mary tells Jeff that Adam is a killer. John tells Jeff that Adam is a killer. Jeff gets angry with Adam
	Enlightenment	Jeff realises that Carlson did not burn out John's barn. Jeff releases Carlson. Jeff says sorry to Carlson.
Rise of the hero		John faces Adam. John demands Adam to leave. Adam refuses to leave the town.
	Redemption	Jeff arrests Adam. Adam shoots Jeff. Carlson shoots Adam. Adam dies. Jeff dies.
Conclusion		Mary says sorry to John. Carlson is freed.

Tabla 6.3: Ejemplo de historia con dos tramas entretejidas mediante la técnica de los vasos comunicantes, tomando las plantillas "El intruso destructor" y "Fausto". Fuente: Concepción et al. (2020)

6.2.4. Análisis de la aproximación evolutiva

La aplicación de programación evolutiva a la tarea de generación de historias multitrama explota aspectos del arquitectura de referencia que no se habían desarrollado en modelos anteriores, como por ejemplo, el uso del revisor de borradores para analizar la población de borradores que se genera durante el proceso. Está descrita en el artículo "Evolutionary Construction of Stories that Combine Several Plot Lines" presentado en el Capítulo 5.

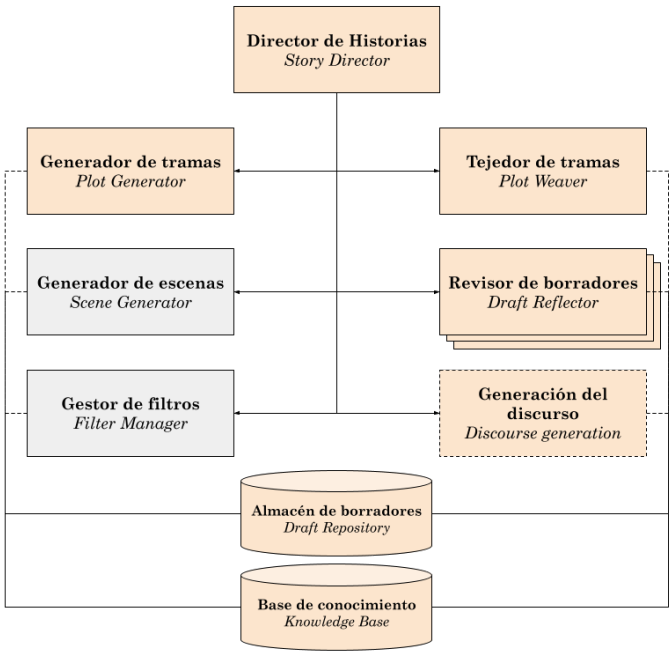


Figura 6.9: Arquitectura de referencia de la solución evolutiva

El proceso de construcción de historias en esta aproximación es el siguiente:

1. El director de historias obtiene del Generador de tramas tantas tramas como se quieran incluir en la historia (este paso es similar al de la versión anterior)

2. El generador de tramas construye las tramas requeridas (las subtramas utilizadas en este caso son las mismas que en el modelo anterior)

3. El director de historias invoca al tejedor de tramas para construir una población inicial de borradores

4. El tejedor de tramas genera tantos borradores como sean necesarios para la población inicial, construyendo cada uno de manera aleatoria.

5. El director de historias aplica los operadores evolutivos (mutación y cruce) a base de enviar individuos seleccionados de la población actual. Hay dos operaciones posibles:

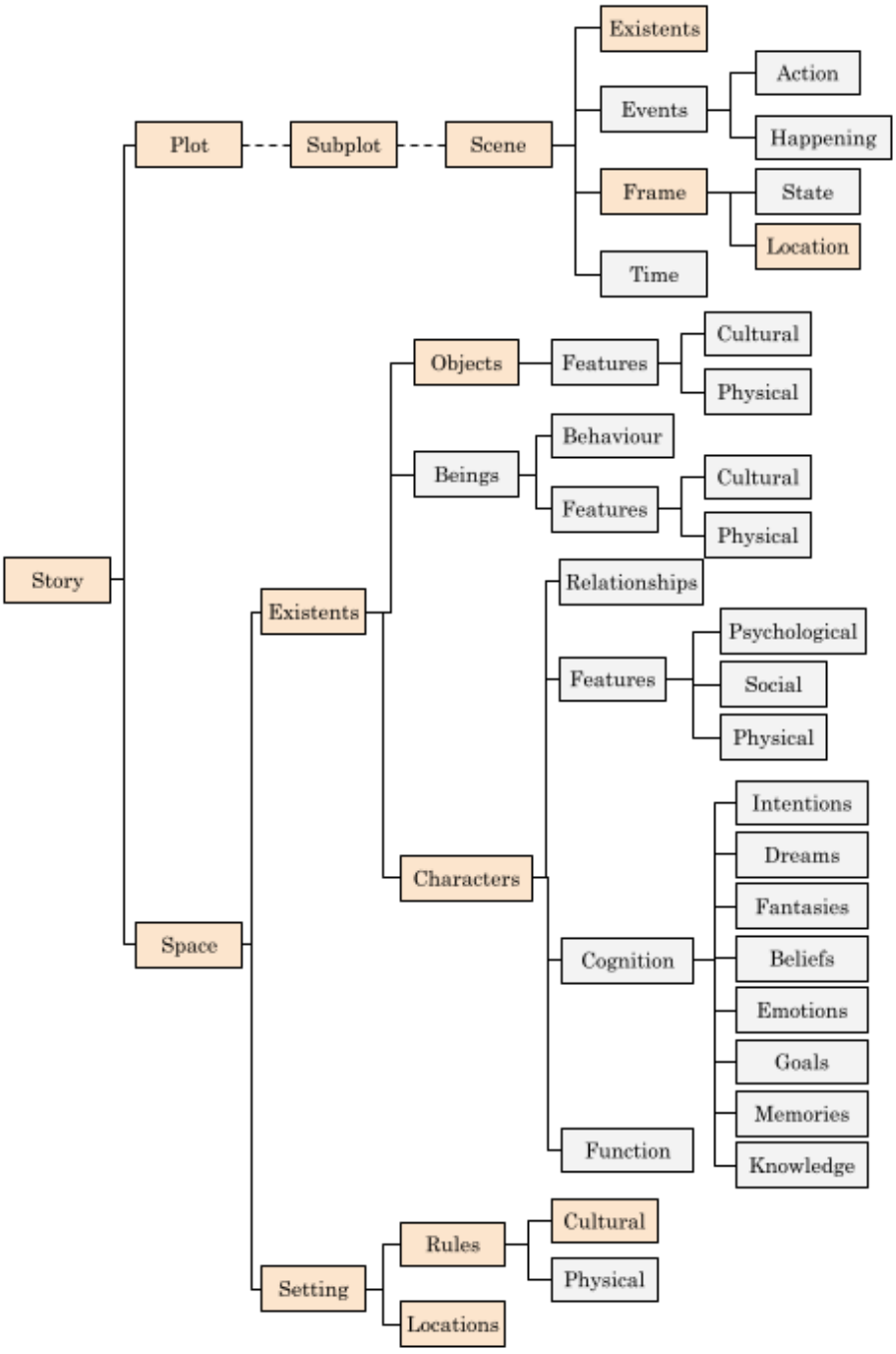


Figura 6.10: Modelo de representación del conocimiento empleado en la aproximación evolutiva

a) El director de historias selecciona un borrador de la población y lo envía a una primera instancia del Revisor de borradores (mutador de borradores)

que devuelve un borrador distinto en el que se han introducido mutaciones de la representación genética (que han alterado o el orden relativo de las escenas o las alineaciones entre personajes de distintas subtramas).

b) El director de historias selecciona dos borradores de la población y los envía a una segunda instancia del Revisor de borradores (cruzador de borradores) que devuelve dos borradores distintos fruto de un cruce de las representaciones genéticas de los dos borradores recibidos (los borradores que resultan de esta operación presentan variaciones con respecto a los originales en el orden relativo de las escenas y en las alineaciones entre personajes de distintas subtramas).

6. El director de historias envía cada individuo de la población acumulada (la original más las variantes generados por los procesos de mutación y cruce) a una tercera instancia del revisor de borradores (evaluador de borradores), que asigna a cada uno una puntuación indicativa de su posible calidad como historia.

7. El director de historias selecciona cuáles de los borradores pasarán a la siguiente generación, y repite el ciclo tantas veces como generaciones se desee desarrollar.

8. Cuando se termina el número de generaciones determinado, el director de historias presenta el borrador mejor puntuado de la última de las generaciones como historia resultante de todo el proceso.

La tabla 6.5 presenta un ejemplo de historia generada con la implementación evolutiva que combina las plantillas "Creación de vida artificial" (CLM) y "Desdoblamiento trágico de la persona" (SPT). Las columnas muestran: el *texto* de la historia desglosado por escenas, la descripción del *plano del discurso* indicando para cada escena: de qué línea argumental procede; la descripción del reparto de personajes, indicando las funciones que desempeñan en las respectivas líneas argumentales; y los nombres que se les han asignado, que reflejan cualquier *fusión de personajes* presente. Las columnas 1 y 2 están alineadas por escenas, la columna 3 se aplica a la historia en su conjunto. El detalle puede consultarse en el artículo original (Gervás et al., 2022).

La tabla 6.6 recoge una serie de ejemplos de estructuras distintas de historia que resultan con distintas combinaciones. Se trata de estructuras de relatos obtenidas aplicando diferentes configuraciones para los rasgos opcionales relativos a la estructura del discurso, utilizando diferentes combinaciones de tramas creadas de las plantillas "Creación de vida artificial" (CLM), "Desdoblamiento trágico de la persona" (SPT), "Fausto" (FA) y "Descenso al inframundo" (DiU). La columna 1 muestra una historia con tramas insertadas, la columna 2 muestra una historia con trama global pero sin trama insertada, la columna 3 muestra una historia sin trama global ni trama insertada. Para facilitar la comprensión de la estructura de los planos del discurso, las escenas de una de las subtramas están resaltadas en negrita. El detalle puede consultarse en el artículo original (Gervás et al., 2022).

A lo largo del proceso, las distintas poblaciones de borradores que se van generando en cada generación se persisten en el Almacén de borradores. La instancia del

Revisor de borradores encargada de evaluar los distintos individuos (evaluador de borradores) recurre a reglas que establecen los méritos relativos de distintos órdenes entre escenas o alineaciones entre personajes. Estas reglas están almacenadas a su vez en la Base de conocimiento.

Con respecto a la representación de conocimiento, este modelo trabaja con los conceptos de la representación común marcados en la figura 6.10. Como se puede ver, al igual que en el caso de INES 2, el proceso de generación se centra en las tramas y subtramas y no entra en el detalle de las escenas, por lo que no interviene el generador de escenas. Otra particularidad de la solución evolutiva es que emplea varias instancias del revisor de borradores para aplicar diversas estrategias de evaluación de la población de borradores que se va generando, tal como recoge la figura 6.9.

6.3. Comparación entre los sistemas presentados

Esta última sección tiene como propósito presentar de forma compendiada las variaciones en cuanto a la implementación de Afanasyev vistas en las secciones anteriores. Para facilitar la identificación de las diferencias, se hace uso de las figuras 6.11 y 6.12.

A partir de las comparativas, se pueden extraer varias conclusiones que se resumen a continuación:

- El modelo de representación del conocimiento es lo suficientemente flexible como para soportar una utilización parcial o incompleta de todos los aspectos. No impone restricciones de partida, y cada sistema puede emplearlo de la forma que mejor le convenga, sin tener que completar todas las facetas de la representación.

- La arquitectura de referencia sólo requiere de la implementación de un conjunto mínimo de microservicios, concretamente, el director de historias y el generador de tramas, y de los dos mecanismos de persistencia (almacén de borradores y base de conocimiento) para operar. El resto de los microservicios son opcionales, e incluso la arquitectura soporta la existencia de varias instanciaciones de un mismo tipo de microservicio, como en el caso del enfoque evolutivo, que implementa varios evaluadores de borradores para aplicar su algorítmica.

- Aún no se ha implementado un sistema que agote toda la capacidad expresiva del modelo de representación. Esta aproximación queda como trabajo futuro.

- Si bien el ecosistema de microservicios de la arquitectura de referencia parece más que suficiente, queda como trabajo futuro analizar la posibilidad de agregar más tipologías de servicios a la arquitectura de referencia que puedan cubrir nuevos casos de estudio.

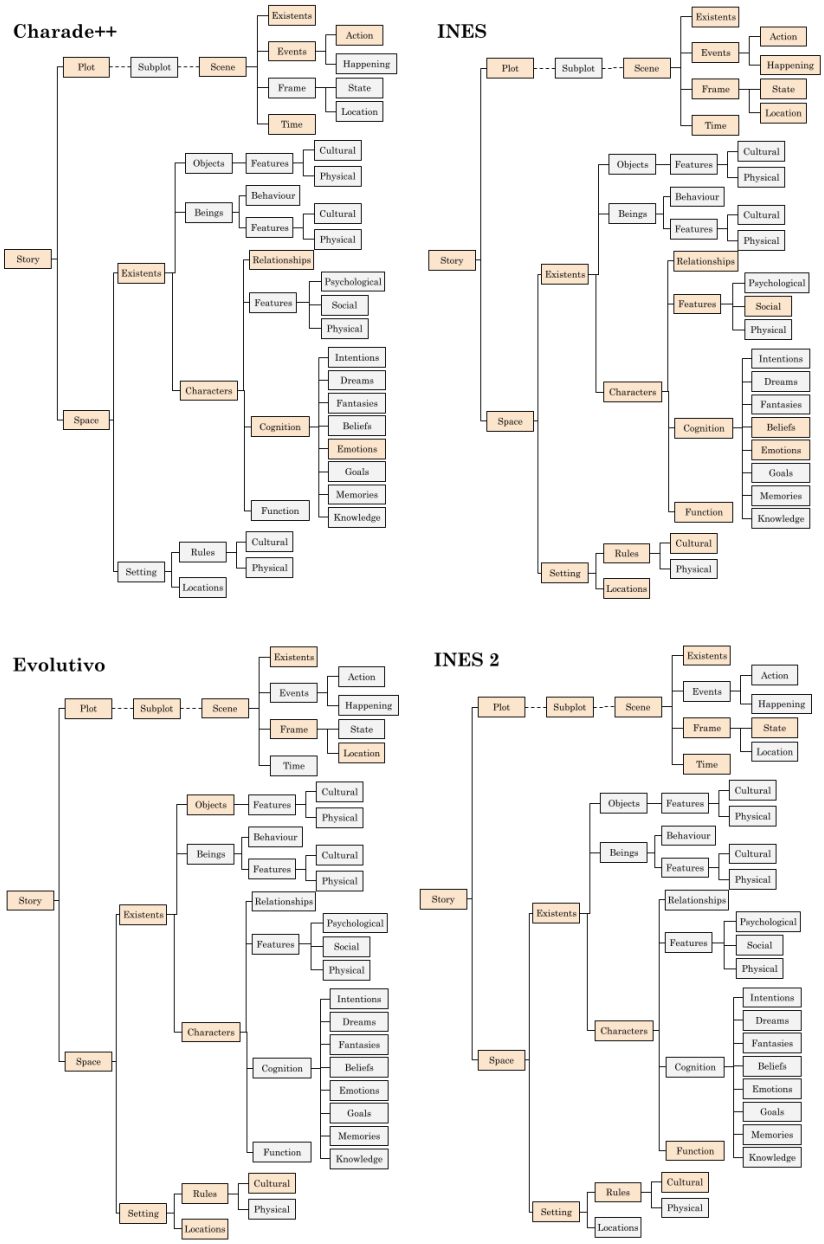


Figura 6.11: Comparativa de la aplicación en los sistemas implementados del modelo general de representación del conocimiento de Afanasyev

6.4. Consideraciones sobre la consecución de los objetivos tecnológicos

6.4.1. Sobre la adopción de microservicios como estilo arquitectónico

Como ya se ha desarrollado en el Capítulo 2, los esfuerzos investigadores en generación automática de historias acumulan ya más de 50 años de resultados. Si

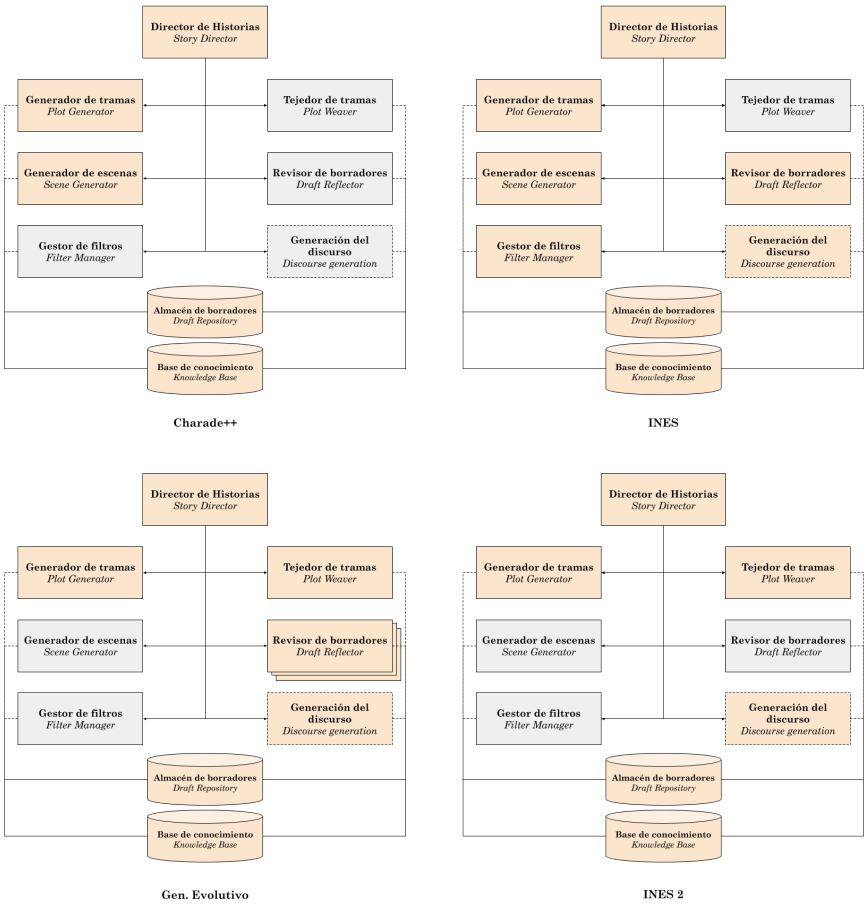


Figura 6.12: Comparativa de la aplicación en los sistemas implementados de la arquitectura de referencia de Afanasyev

bien los sistemas generadores han experimentado una continua mejora a lo largo de la evolución de la disciplina, hay una constante que se ha mantenido mayoritariamente a lo largo de todo este tiempo: un diseño arquitectónico rígido, orientado a objetivos de generación muy concretos. Salvo algunas excepciones ya mencionadas en el Capítulo 2, como el trabajo de Montfort et al. (2013), rara vez se han aplicado enfoques más flexibles, en los que los componentes del sistema pudieran ser fácilmente extendidos o reemplazados en sus implementaciones.

Uno de los objetivos de la presente investigación, referido en el Capítulo 1, es desarrollar una arquitectura de referencia que proporcione flexibilidad para la creación de sistemas de generación de historias con múltiples tramas. Para alcanzar dicho objetivo, se ha trabajado en una arquitectura de referencia y esta se ha puesto a prueba en varios sistemas que la implementan.

Las razones para haber adoptado un diseño basado en microservicios se debe a que estos proporcionan varios atributos de calidad deseables en un sistema software (Kazman et al., 1999), como la flexibilidad (*Flexibility*), la capacidad de evolucionar (*Evolvability*) y la capacidad de reemplazo de la implementación de una parte del

sistema por otra (*Replaceability*). Estas características son especialmente deseables para el propósito deseado con la arquitectura, pues permiten realizar cambios y extensiones de los diversos componentes con vistas a refinar el proceso de generación y, particularmente, con el desarrollo del proceso de generación de historias con múltiples tramas. Como se puede ver en los artículos correspondientes al Capítulo 4.

A partir de este objetivo, y tomándolo en consideración para su traslación como *driver* arquitectónico (Wojcik et al., 2006), se han identificado las siguientes cualidades del sistema como primordiales:

- Flexibilidad. La flexibilidad es un atributo de los productos de software que pueden adaptarse fácilmente a los cambios futuros.
- Reemplazabilidad. Atributo que mide el grado de eficacia con el que un producto de software puede sustituir a otra solución de software diseñada para el mismo fin.
- Capacidad de evolucionar. Atributo que expresa la facilidad con la que el sistema puede evolucionar para agregar nuevas capacidades.
- Facilidad de prueba (*Testability*). Atributo que expresa la facilidad con la cual se elaboran pruebas efectivas para el sistema.
- Modificabilidad. Atributo que mide el coste de realizar cambios en el sistema.
- Interoperabilidad. Atributo acerca de la facilidad del sistema para intercambiar información con otros sistemas mediante interfaces.

Estas cualidades deseadas, unidas al contexto actual de desarrollo de software, en el que priman los estilos arquitectónicos distribuidos precisamente por su flexibilidad, han guiado la decisión de adoptar una arquitectura basada en microservicios. Un microservicio es, en palabras de Wolff (2016), la expresión del Principio de Responsabilidad Única a nivel arquitectónico. Esto expresa que un microservicio es un componente en el que prima la consistencia a nivel arquitectónico. Cada microservicio es responsable de una función específica en un ecosistema más amplio. Este es el criterio que ha guiado el diseño de la arquitectura de referencia de Afanasyev: identificar las funciones principales en el proceso de generación, consistentes con el modelo de representación del conocimiento definido, y trasladarlas a microservicios especializados. Esto se puede ver en el servicio generador de tramas y su correspondiente expresión en el modelo de conocimiento, o en el generador de escenas, o en el tejedor de tramas, por mencionar algunos ejemplos.

El estilo arquitectónico basado en microservicios presenta además otras ventajas (Newman, 2015; Wolff, 2016): cada servicio está desacoplado del resto, por lo que la comunicación entre los mismos se ciñe estrictamente a los contratos definidos en sus API. Esto simplifica enormemente la posibilidad de reemplazar la implementación de un microservicio por otra, variando su comportamiento interno pero preservando el contrato expresado en la API, sin que afecte al conjunto. Con este mecanismo se pueden probar diferentes técnicas, bien compaginándolas, bien reemplazándolas, en el ecosistema de microservicios.

A partir de las razones expuestas, se puede concluir que la adopción de un estilo arquitectónico basado en microservicios aporta una solución a las necesidades y objetivos expuestos, cubriendo a nivel técnico los resultados buscados.

6.4.2. Sobre el modelo de representación

La información necesaria para generar historias depende de una serie de factores. Uno de estos factores clave es la arquitectura del sistema. Los componentes que participan en el proceso de generación condicionan la estructura del conocimiento. Por ejemplo, en el caso de los sistemas de generación de historias construidos sobre planificadores, es necesario mantener el conocimiento relativo a los estados, las precondiciones, las acciones, los efectos de las acciones, etc. Tal como se refleja en el Capítulo 2, los generadores de historias basados en gramáticas requieren una representación completa de las reglas aplicables para crear sus historias; los generadores de historias basados en la simulación requieren una tipificación detallada de los personajes, sus relaciones y unos operadores que modifican el comportamiento y las relaciones de los personajes; y así con el resto de familias de generadores.

Por encima de estas particularidades existe un elemento común a todos los sistemas de narración que puede ser intercambiado: la historia, que es el producto resultante del proceso de generación. Dependiendo de cada modelo de generación, la historia puede ser generada en una única iteración o proceso, o irse modelando a partir de diversas pasadas por los componentes. En cualquier caso, si el producto que se genera en cada componente de un sistema generador, ya sea una historia completa o un bosquejo de historia, puede ser traducido a una representación agnóstica, susceptible de ser intercambiada y comprendida por otro sistema con una arquitectura y operación diferentes, los resultados producidos por el uno pueden ser enriquecidos por el otro. Este es un beneficio derivado de emplear un modelo de representación del conocimiento compartido, como el propuesto en Afanasyev. Se busca permitir la colaboración entre sistemas generadores, o entre algunos de sus componentes, de forma que pueda articularse un proceso combinado de generación.

El modelo de representación propuesto en la presente tesis se centra en el conocimiento que está directamente relacionado con la historia, en lugar del relacionado con el proceso de generación, que sería difícil de exportar entre diferentes sistemas. Es por este motivo que, como se puede ver en la sección anterior, es posible crear sistemas generadores que, aprovechando de formas diferentes el modelo común de representación, logran cubrir sus objetivos de manera simple y flexible.

Descent into Underworld	**Creation of Life**	**Faust**	**Actions**
Happy lovers			Hawa loves Seth. Seth loves Hawa. Korr hates Hawa. Korr hates Seth. Hawa has a magic sword. Hawa has a magic wand.
Lost			Korr spells a cast against Seth. Seth is turned into stone.
Mourning and quest			Hawa is sad. Hawa travels to Korr's kingdom.
	Prometheus Dream		Hawa needs a minion.
	Creation of new life		Hawa studies how to create life with magic. Hawa learns how to create life using dark magic. Hawa creates a minion using dark magic. Hawa names the minion Adam.
	The being complies		Hawa orders Adam to fight a dragon. Adam defeats a dragon.
	Rebellion and escape		Hawa orders Adam to fight a troll. Adam refuses to fight. Adam flies.
	Infringement		The being infringes the human laws and moral principles
	Quest for the creation		Adam goes to a city. Hawa pursues Adam. Adam enters into Sam's store. Adam steals food and clothes. Sam discovers Adam. Adam kills Sam.
	Duel with creation		Hawa finds Adam. Adam fights back Hawa. Hawa cast a spell on Adam. Adam disappears.
Deal and brief reunion			Hawa arrives at Korr's castle. Hawa asks for a deal. Korr asks Hawa to give him her magic sword. Hawa accepts. Korr gives Hawa a potion to save Seth. Korr asks Hawa not to open the bottle before reaching Seth. Hawa travels back her country.
Infringement			Hawa does not trust Korr. Hawa opens the potion before reaching Seth. The potion disappears.
Metamorphosis			Hawa has lost Adam. Hawa is sad.
		Frustration	Hawa is sad. Hawa regrets her life. Hawa studies dark magic.
		Temptation	Hawa invokes an evil demon. Thrall appears. Thrall offers Hawa to be a queen.
		Pact with evil	Hawa makes a deal with Thrall. Hawa becomes the queen.
		Evil actions	Thrall persuades Hawa to raise the taxes. Hawa raise the taxes. Thrall persuades Hawa to punish the debtors. Hawa orders to imprison the debtors.
		Enlightenment	Lem is a friend of Hawa. Lem tells Hawa that her vassals are suffering. Hawa realises that she is acting badly.
		Redemption	Hawa spells a cast on Thrall. Thrall curse Hawa. Thrall dies. Hawa dies.

Tabla 6.4: Ejemplo de historia con dos tramas entretejidas mediante la técnica de las cajas chinas empleando las plantillas "Descenso al inframundo", "Creación de vida artificial" y "Fausto". Fuente: Concepción et al. (2020)

Text	#	Plot	Scene name	Variable	Role	Name
A is a doctor researching life improving serum. A has dedicated his whole life to science and has no friends or family.	0	SPT	Introduction	CLM-0	Frankenstein	C
A accidentally creates a serum that changes him into B, an evil twin of himself.	1	SPT	Splitting	CLM-1	Creature	D
C is a German nobleman. C lives in a big castle. C is sad. C 's wife died recently. C loves his dead wife. C dreams of bringing to life his wife.	2	CLM	Conception of Idea	CLM-2	-	E
C studies book of magic. C discovers a spell to bring to life a portrait of his wife. C brings to life D , the portrait of his wife.	3	CLM	Creation of new life	SPT-0	DrJekyll	A
C talks every evening with D. C tells D that C loves D. D is kind to C.	4	CLM	The being complies with its master's will	SPT-1	MrHyde	B
C goes to a ball. C meets E in a ball. C falls in love with E. E falls in love with C. C tells D C loves E.	5	CLM	Rebellion and escape	SPT-2	-	C
E visits C. D is jealous of E. D takes E and disappears.	6	CLM	Infringement by creature			
C searches for D across the castle.	7	CLM	Quest for the creature			
D tells C that C must defeat D to save E. C burns the portrait. D is destroyed and E appears magically. C and E are happy.	8	CLM	Death of the creature			
B brutally kills some friends of A.	9	SPT	Entanglements			
C realises that B is an evil version of A. A confesses to C what has happened.	10	SPT	Discovery			
A decides to stop taking the serum.	11	SPT	Unravelling			
A transforms involuntary into B. B goes on a rampage. Having lost control, A decides to commit suicide, killing B in the process.	12	SPT	Denouement			

Tabla 6.5: Ejemplo de historia generada por la implementación evolutiva. Fuente: Gervás et al. (2022)

#	Plot	Scene name	#	Plot	Scene name	#	Plot	Scene name
0	SPT	Introduction	0	SPT	Introduction	0	DiU	Lovers Initial Happiness
1	SPT	Splitting	1	**FA**	Frustrated character regrets his life	1	DiU	Lost
2	**CLM**	Conception of Idea	2	SPT	Splitting	2	DiU	Mourn and quest for the beloved
3	**CLM**	Creation of new life	3	**FA**	Temptation	3	**FA**	Frustrated character regrets his life
4	**CLM**	The being complies with its master's will	4	**FA**	Pact with evil	4	**FA**	Temptation
5	**CLM**	Rebellion and escape	5	SPT	Entanglements	5	**FA**	Pact with evil
6	**CLM**	Infringement by creature	6	**FA**	Evil actions	6	DiU	Deal and brief reunion
7	**CLM**	Quest for the creature	7	**FA**	Enlightenment	7	**FA**	Evil actions
8	**CLM**	Death of the creature	8	SPT	Discovery	8	DiU	Infringement of compromise
9	SPT	Entanglements	9	**FA**	Redemption	9	DiU	Metamorphosis
10	SPT	Discovery	10	SPT	Unravelling	10	**FA**	Enlightenment
11	SPT	Unravelling	11	SPT	Denouement	11	**FA**	Redemption
12	SPT	Denouement						

Tabla 6.6: Estructuras diferentes de historia producto de distintas combinaciones en el proceso evolutivo. Fuente: Gervás et al. (2022)

Capítulo 7

Conclusiones y trabajo futuro

"Life is the art of drawing sufficient conclusions from insufficient premises"
— Samuel Butler

7.1. Conclusiones

A lo largo del trabajo presentado existe una línea de guía que reúne los diferentes objetivos específicos declarados en el Capítulo 1. A modo de recordatorio, se establecieron los siguientes **objetivos científicos**:

- Objetivo C1: Desarrollar un modelo de representación del conocimiento que permita representar historias con múltiples tramas
- Objetivo C2: Desarrollar una arquitectura de referencia que proporcione flexibilidad para la creación de sistemas de generación de historias con múltiples tramas
- Objetivo C3: Diseñar mecanismos para generar automáticamente historias con múltiples tramas

Y estrechamente relacionados con dichos objetivos científicos, se expresaban los siguientes **objetivos tecnológicos**:

- Objetivo T1: Desarrollar sistemas que permitan poner a prueba la arquitectura diseñada
- Ojetivo T2: Desarrollar sistemas que implementen los mecanismos de generación de historias con múltiples tramas

En línea con dichos objetivos, la investigación desarrollada ha puesto foco en tres áreas principales: desarrollo y prueba de un modelo de representación del conocimiento que diera soporte a la generación de historias mutiltrama, desarrollo de heurísticas para generar automáticamente historias multitrama y desarrollo y prueba

de una arquitectura que permitiera implementar dichas heurísticas integrando también el modelo de representación definido. Las conclusiones se agrupan en estas tres líneas para facilitar su exposición.

7.1.1. Conclusiones relativas al modelo de representación del conocimiento

El modelo de representación del conocimiento que se presenta a lo largo de este trabajo se basa en el análisis de los modelos de represntación del conocimiento empleados por un buen número de sistemas de generación automática existentes. Trata de establecer una línea de base para facilitar el soporte a historias con múltiples tramas, y al mismo tiempo, a hacer posible la generación de historias más ricas y de mayor calidad aplicando un criterio maximalista a la hora de considerar todos los elementos de conocimiento con potencial valor para la generación.

Para no vincular el modelo a un proceso específico de generación de historias, el modelo propuesto se centra en el producto sobre el que trabaja un sistema generador, es decir, en la historia. Este enfoque, que también se ha aplicado en la arquitectura de referencia, que trata de centrarse en etapas genéricas del proceso, parece ser el más conveniente para soportar la mayor flexibilidad posible.

Este modelo de representación del conocimiento está complementado con un modelo de arquitectura que permite el desarrollo de un proceso de generación que soporta las historias multi-trama.

Además de lo expuesto, cabe destacar las siguientes conclusiones a partir de la investigación:

- Resulta importante representar el argumento de una narración más allá del planteamiento clásico de una secuencia de eventos, utilizando distintos tipos de granularidad (jerarquía de representación de conocimiento sobre narrativa) que permitan tanto agrupar eventos en unidades más grandes de representación que describan escenas, como agrupar escenas en unidades de representación más grandes todavía que describan subtramas, como estructurar subtramas en unidades específicas de combinación que describan historias completas con un grado elevado de complejidad

- De cara a trabajar con estas unidades intermedias es muy importante establecer un nivel de representación que permita etiquetar las distintas variedades de participación de los personajes de la historia (roles de personaje en una escena, roles de personaje en una subtrama, roles de personaje en una trama)

- El desarrollo de procesos de combinación de tramas para generar historias complejas tiene tener en cuenta estos niveles adicionales de representación que no se habían contemplado de manera explícita en trabajos previos sobre el tema

- Los prototipos desarrollados constituyen ejemplos de solución de problemas específicos que forman parte del proceso de generar una narrativa compleja, pero que necesariamente deben considerarse primeros pasos en un camino cuyo

recorrido principal está todavía por recorrer (puesto que no se han cubierto en ellos nada más que un subconjunto pequeño de los aspectos que participan y determinan la complejidad de una narrativa totalmente desarrollada)

7.1.2. Conclusiones relativas a la arquitectura de referencia

El objetivo principal del proceso de diseño de diseño de la arquitectura de referencia es permitir la construcción de sistemas generadores de historias que, ajustándose al marco provisto por esta, puedan generar historias con múltiples tramas. Además, la arquitectura de referencia tiene que ser compatible con el modelo de representación del conocimiento que también se persigue en los objetivos de la investigación.

El propósito de desarrollar los sistemas generadores, Charade++, INES, INES 2 y el generador evolutivo, era demostrar la idoneidad del marco Afanasyev para construir o incluso reconstruir cualquier aplicación de generación de historias. En este sentido, Afanasyev proporciona una base sólida para el desarrollo de sistemas de generación consistentes con los objetivos ya mencionados. A lo largo de este trabajo se han mostrado las principales capacidades de los sistemas generados, como implementar un ecosistema flexible y modular y presentar arquitectura bien fundamentada de acuerdo con la representación del conocimiento.

La inclusión de componentes en la arquitectura procedente de Charade ha aportado riqueza en la forma de generar historias, generando un ecosistema bastante diferente de Charade, pero la fuerte consistencia de la representación del conocimiento y el modelo de comunicación de la arquitectura han facilitado el mantenimiento de una intercomunicación consistente y coherente entre todos los servicios.

Además de estos aspectos, hay que destacar también como parte de las conclusiones:

- La introducción de la jerarquía en la representación del conocimiento sobre narrativa presenta ventajas operativas de cara al desarrollo de sistemas computacionales de generación, por cuanto que permite el aprovechamiento de sistemas o soluciones ya existentes como módulos que producen material a los niveles de representación ya considerados en esfuerzos anteriores (generación de discurso textual a partir de representación conceptual de eventos, generación de secuencias de escenas como tramas simples); y permite el desarrollo de módulos específicos para generar soluciones a los niveles de representación que se han añadido a la jerarquía, sea como combinación de elementos generados por otros módulos (tejedor de tramas) o como adaptación de procesos de generación de historias completas para construir unidades ligeramente más pequeñas (generador de escenas)

- La introducción del concepto de revisor de borradores como un tipo de módulo explícito dentro de la arquitectura de referencia permite el desarrollo de soluciones a la tarea de generación que son, por un lado, identificables como emulaciones computacionales de procesos que se observan en los autores humanos (reescritura iterativa de borradores); y, por otro, permiten la implementación

en el contexto de la arquitectura propuesta de técnicas de inteligencia artificial de relativa complejidad (como los procesos de programación evolutiva).

7.1.3. Conclusiones relativas a los mecanismos de generación de historias con múltiples tramas

La exploración llevada a cabo en esta investigación ha identificado un conjunto de conceptos relevantes para la tarea de entrelazar tramas, que han quedado reflejados en el modelo de representación de historias; ha establecido algunas líneas de base computacionales para evaluar las estrategias de entretejido de tramas, y ha elevado indicios relevantes en cuanto al impacto de los diversos tipos de entretejido de tramas en la impresión que producen en un lector. Las técnicas básicas de tejido de tramas descritas han demostrado además su viabilidad computacional y su capacidad para generar efectivamente contenido con sentido.

Las dos técnicas avanzadas de entrelazar tramas que se proponen, los **vasos comunicantes** y las **cajas chinas**, proceden del contexto literario. La investigación ha demostrado la viabilidad de trasladar estas técnicas de creatividad de la literatura hecha por humanos a la narración automática. Aunque los algoritmos se han desarrollado para el sistema objetivo, el marco de referencia de Afanasyev, el modelo conceptual en el que se basan es lo suficientemente genérico como para ser exportado a otros sistemas de narración. Por otro lado, se ha podido constatar que precisamente dicho marco, contando con la arquitectura de referencia y la representación del conocimiento, es un soporte efectivo para la implementación de las mencionadas heurísticas.

El mecanismo de comprobación de consistencia presentado en los artículos, aunque todavía tiene margen de mejora, ayuda a proporcionar una continuación razonable entre escenas de diferentes líneas argumentales.

El uso de una estrategia de trama aleatoria como línea de base también se ha revelado como una herramienta muy interesante para comparar la calidad de otras técnicas de tejido de tramas. La estrategia aleatoria es, por tanto, una línea de base razonable que establece un umbral que otras estrategias tendrán que superar. Las futuras investigaciones sobre la realización de un estudio orientado al usuario para analizar la calidad de los relatos utilizarán probablemente esta línea de base.

Así pues, la reflexión llevada a cabo sobre el proceso de generación de historias con múltiples subtramas desde el punto de vista computacional conduce a las siguientes conclusiones:

- El proceso de construcción de una única trama está sujeto a restricciones distintas del encadenamiento progresivo de un número cada vez mayor de escenas; por lo que se hace relevante que, además de existir continuidad entre escenas sucesivas, exista una progresión de la trama hacia una conclusión general que cierre de manera adecuada la acumulación de escenas generadas.

- El proceso de construcción de una historia con múltiples subtramas difiere del proceso de construcción de una historia con una única trama; en tanto que cada subtrama requiere un proceso equivalente al de una trama única de buscar y

construir un cierre específico de los temas introducidas en las escenas específicas de la subtrama

- Finalmente, y por las razones enumeradas anteriormente, es igualmente esencial trabajar con una representación de la historia que incluya el concepto de subtrama como unidad explícita de articulación de la narrativa, como trabajar con una arquitectura que permita el desarrollo y la combinación de módulos que traten explícitamente estas distintas tareas de construcción.

7.2. Líneas de trabajo futuro

Ciertamente, el trabajo de investigación ha abierto nuevas e interesantes líneas de trabajo futuras. Las que se recogen en esta sección son una muestra de aquellas con un mayor potencial o relevancia en términos investigadores.

7.2.1. Un modelo arquitectónico basado en eventos

Después de haber considerado los objetivos de la tesis y los *drivers* arquitectónicos descritos en este documento, una línea de evolución arquitectónica que surge de forma natural sería adoptar un estilo de microservicios basado en eventos. Para ello, sería necesario adaptar el ecosistema construido para que el intercambio entre componentes girara alrededor de una infraestructura orientada a mensajes que centralice el intercambio de eventos entre los diferentes servicios.

Este enfoque para construir una arquitectura con una mejor escalabilidad y facilidad de extensión supondría una evolución del modelo de referencia Afanasyev. En este caso, el objetivo sería reducir la necesidad de un intercambio continuo de llamadas síncronas basadas en REST con el director de historias como componente centralizador. Su papel podría ser reemplazado por un nuevo componente, el **editor de historias** (*Story Editor*), que, de forma asíncrona, estaría observando la evolución de los borradores en el almacén de borradores y decidiría cuándo un borrador debe ser publicado como historia terminada. En líneas generales, la estructura de componentes dentro del proceso de generación se mantendría, pero reemplazando la gobernanza síncrona, dirigida y centralizada, por un mecanismo asíncrono y no intervencionista.

7.2.1.1. El bus de borradores (Draft Bus)

El **bus de borradores** sería el eje central de un ecosistema arquitectónico ampliado. Proporcionaría una infraestructura basada en eventos tanto para la persistencia (como sustituto del almacén de borradores) como para compartir las actualizaciones relacionadas con el proceso de generación de historias. Desde un punto de vista técnico, puede implementarse utilizando un middleware orientado a mensajes (Curry, 2004). Los mensajes intercambiados en esta infraestructura serían principalmente eventos cuyo contenido se ajustaría al modelo de representación del conocimiento ya existente.

7.2.1.2. El mezclador de borradores

El **mezclador de borradores** (*Draft Mixer*) sería un nuevo componente de la arquitectura de referencia Afanasyev que encajaría bien con la línea anterior. Este componente recolectaría los eventos relacionados con un proyecto de generación de historia (compuesto por uno o varios borradores en definición), tomaría los diferentes borradores del bus de borradores y los combinaría para generar una historia con múltiples tramas. El comportamiento de este servicio sería sin estado: cada vez que recibiera un evento conteniendo un nuevo borrador, consultaría la lista de proyectos en curso de generación para recuperar el borrador asociado a dicho proyecto y fusionaría el dicho borrador con el nuevo incluido en el evento. Tras fusionar ambas historias, el Mezclador de Borradores persiste la resultante en el Repositorio de Proyectos como versión actual, a la espera de recibir nuevos borradores con los que fusionar, y publica un evento correspondiente en el bus de borradores.

7.2.2. Ampliación de las técnicas multitrama

El mezclador de tramas podría ampliar el rango de heurísticas que soporta actualmente. Una línea de investigación interesante en este sentido sería la inclusión de una estrategia "Multiperspectiva", en la que las tramas son básicamente la misma secuencia de acontecimientos pero retratada desde la perspectiva de diferentes personajes. La investigación y el desarrollo de una heurística basada en esta aproximación podría dar lugar a resultados narrativos prometedores.

7.2.3. Creación de un entorno multi-sistema

Aprovechando la modularidad de la arquitectura de referencia y la flexibilidad del propio modelo de representación del conocimiento, sería posible establecer una línea futura que buscara integrar componentes de la arquitectura implementados a partir de partes de otros sistemas generadores, aumentando la variedad de las historias y fomentando la colaboración.

Algunos de los puntos esenciales que habrá que evaluar son los relacionados con la consistencia del conocimiento entre sistemas, la posible degradación de la semántica y la pérdida de información relevante durante la comunicación entre los sistemas participantes. Seguramente, el trabajo futuro sobre el ecosistema de narración propuesto permitirá un estudio completo de estas preocupaciones, así como la mejora del modelo para garantizar que el resultado final cubra todos los objetivos especificados.

Bibliografía

Ordenar bibliotecas es ejercer, de un modo modesto y silencioso, el arte de la crítica

Jorge Luis Borges

Aamodt, A. y Plaza, E. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, vol. 7(1), páginas 39–59, 1994.

Alhussain, A. I. y Azmi, A. M. Automatic story generation: a survey of approaches. *ACM Computing Surveys (CSUR)*, vol. 54(5), páginas 1–38, 2021.

Antoniou, G. y Harmelen, F. v. Web ontology language: Owl. En *Handbook on ontologies*, páginas 67–92. Springer, 2004.

Aristóteles. Poética, ed. trilingüe de valentín garcía yebra. *Madrid, Gredos*, vol. 19902, 1974.

Balló, J. y Pérez, X. *La semilla inmortal: los argumentos universales en el cine*. Ed. Anagrama, 2007.

Barthes, R. *S/Z: an essay*. Siglo XXI, 1980.

Barzilay, R. y Lapata, M. Modeling local coherence: An entity-based approach. *Computational Linguistics*, vol. 34(1), páginas 1–34, 2008.

Bass, L., Clements, P. y Kazman, R. *Software architecture in practice*. Addison-Wesley Professional, 2003.

Binnick, R. I. An application of an extended generative semantic model of language to man-machine interaction. En *Proceedings of the 1969 conference on Computational linguistics*, páginas 1–34. Association for Computational Linguistics, 1969.

Black, J. B. y Bower, G. H. Story understanding as problem-solving. *Poetics*, vol. 9(1-3), páginas 223–250, 1980.

Black, J. B. y Wilensky, R. An evaluation of story grammars. *Cognitive science*, vol. 3(3), páginas 213–229, 1979.

Bringsjord, S. y Ferrucci, D. *Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine*. Psychology Press, 1999.

Bringsjord, S. y Ferrucci, D. A. Artificial intelligence and literary creativity: Inside the mind of brutus, a storytelling machine. *Computational Linguistics*, vol. 26(4), 2000.

Brinke, H. *Hide and sneak-Perceptions in The Virtual Storyteller*. Proyecto Fin de Carrera, Electrical Engineering, Mathematics and Computer Science, 2014.

Carberry, S. Techniques for plan recognition. *User modeling and user-adapted interaction*, vol. 11(1), páginas 31–48, 2001.

Carbonell, J. G. Subjective understanding: Computer models of belief systems. Informe técnico, DTIC Document, 1979.

de Cervantes Saavedra, M. *El ingenioso hidalgo Don Quijote de la Mancha*, vol. 2. Espasa-Calpe (Ed. 2005), 1605.

Chang, H.-M. y Soo, V.-W. Planning to influence other characters in agent-based narratives. En *Integrating Technologies for Interactive Stories Workshop, International Conference on Intelligent Technologies for Interactive Entertainment*, páginas 12–17. 2008.

Chatman, S. B. *Story and discourse: Narrative structure in fiction and film*. Cornell University Press, 1980.

Chi, P.-Y. y Lieberman, H. Raconteur: from intent to stories. En *Proceedings of the 15th international conference on Intelligent user interfaces*, páginas 301–304. 2010.

Chi, P.-Y. y Lieberman, H. Raconteur: Integrating authored and real-time social media. En *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, páginas 3165–3168. 2011.

Clements, P., Kazman, R., Klein, M. et al. *Evaluating software architectures*. Tsinghua University Press Beijing, 2003.

Cohen, P. R. y Feigenbaum, E. A. *The handbook of artificial intelligence*, vol. 3. Butterworth-Heinemann, 2014.

Concepción, E., Gervás, P. y Méndez, G. Mining knowledge in storytelling systems for narrative generation. En *Proceedings of the INLG 2016 Workshop on Computational Creativity in Natural Language Generation*, páginas 41–50. 2016a.

Concepción, E., Gervás, P. y Méndez, G. An api-based approach to co-creation in automatic storytelling. En *6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017*. 2017a.

Concepción, E., Gervás, P. y Méndez, G. A common model for representing stories in automatic storytelling. En *6th International Workshop on Computational Creativity, Concept Invention, and General Intelligence. C3GI 2017*. 2017b.

Concepción, E., Gervás, P. y Méndez, G. A microservice-based architecture for story generation. En *Microservices 2017*. 2017c.

Concepción, E., Gervás, P. y Méndez, G. Afanasyev: A collaborative architectural model for automatic story generation. En *5th AISB Symposium on Computational Creativity. AISB 2018*. 2018a.

Concepción, E., Gervás, P. y Méndez, G. Ines: A reconstruction of the charade storytelling system using the afanasyev framework. En *Ninth International Conference on Computational Creativity, ICCC 2018*. Salamanca, Spain, 2018b.

Concepción, E., Gervás, P. y Méndez, G. Evolving the ines story generation system: from single to multiple plot lines. En *10th International Conference on Computational Creativity (ICCC 2019)*. UNC Charlotte, North Carolina, USA, 2019.

Concepción, E., Gervás, P. y Méndez, G. Exploring baselines for combining full plots into multiple-plot stories. *New Generation Computing*, vol. 38(4), páginas 593–633, 2020.

Concepción, E., Gervás, P., Méndez, G. y León, C. Using cnl for knowledge elicitation and exchange across story generation systems. En *International Workshop on Controlled Natural Language*, páginas 81–91. Springer, 2016b.

Concepción, E., Méndez, G., Gervás, P. y León, C. A challenge proposal for narrative generation using cnls. En *Proceedings of the 9th International Natural Language Generation conference*, páginas 171–173. 2016c.

Cortázar, J. *Rayuela*. Editorial Sudamericana, Buenos Aires, 1963.

Crockford, D. The application/json media type for javascript object notation (json). 2006.

Curry, E. Message-oriented middleware. *Middleware for communications*, páginas 1–28, 2004.

Dehn, N. Memory in story invention. En *Proceedings of the third annual conference of the cognitive science society*, páginas 213–215. 1981a.

Dehn, N. Story generation after tale-spin. En *IJCAI*, vol. 81, páginas 16–18. 1981b.

Donen, S. Charade. 1963.

Erl, T. *Service-oriented architecture: a field guide to integrating XML and web services*. Prentice Hall PTR, 2004.

Faas, S. Virtual storyteller: an approach to computational storytelling. *Unpublished master's thesis, University of Twente, Department of Electrical Engineering, Mathematics and Computer Science*, 2002.

Fay, M. P. *Driving story generation with learnable character models*. Tesis Doctoral, Massachusetts Institute of Technology, 2014.

Fielding, R. T. *Architectural styles and the design of network-based software architectures*. Tesis Doctoral, University of California, Irvine, 2000.

Flaubert, G. *Madame Bovary*. Michel Lévy, 1857.

Fuchs, N. E. y Schwitter, R. Specifying logic programs in controlled natural language. *arXiv preprint cmp-lg/9507009*, 1995.

Gat, I. y Succi, G. A survey of the api economy. *Cut. Consort*, 2013.

Gatt, A. y Reiter, E. Simplenlg: A realisation engine for practical applications. En *Proceedings of the 12th European Workshop on Natural Language Generation*, páginas 90–93. Association for Computational Linguistics, 2009.

Genette, G. *Narrative discourse : an essay in method*. Cornell University Press, 1980.

Gervás, P. Story generator algorithms. En *The Living Handbook of Narratology*. Hamburg University Press, 2012.

Gervás, P. Propp's morphology of the folk tale as a grammar for generation. En *OASIcs-OpenAccess Series in Informatics*, vol. 32. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.

Gervás, P. Computational drafting of plot structures for russian folk tales. *Cognitive computation*, vol. 8(2), páginas 187–203, 2016.

Gervás, P. Storifying observed events: Could i dress this up as a story? En *5th AISB Symposium on Computational Creativity*. AISB, AISB, University of Liverpool, UK, 2018.

Gervás, P. Generating a search space of acceptable narrative plots. En *10th International Conference on Computational Creativity (ICCC 2019)*. UNC Charlotte, North Carolina, USA, 2019.

Gervás, P., Concepción, E., León, C., Méndez, G. y Delatorre, P. The long path to narrative generation. *IBM Journal of Research and Development*, vol. 63(1), páginas 8–1, 2019.

Gervás, P., Concepción, E. y Méndez, G. Assessing multiplot stories: From formative analysis to computational metrics. En *ICCC*, páginas 92–96. 2021.

Gervás, P., Concepción, E. y Méndez, G. Evolutionary construction of stories that combine several plot lines. En *International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar)*, páginas 68–83. Springer, 2022.

Gervás, P. y León, C. The need for multi-aspectual representation of narratives in modelling their creative process. En *5th Workshop on Computational Models of Narrative*, OASIcs-OpenAccess Series in Informatics. 2014.

Ghallab, M., Nau, D. y Traverso, P. *Automated Planning: theory and practice*. Elsevier, 2004.

Greimas, A. J. *Structural semantics: An attempt at a method*. University of Nebraska Press, 1983.

Han, J., Haihong, E., Le, G. y Du, J. Survey on nosql database. En *Pervasive computing and applications (ICPCA), 2011 6th international conference on*, páginas 363–366. IEEE, 2011.

Harrell, D. F. Walking blues changes undersea: Imaginative narrative in interactive poetry generation with the griot system. En *AAAI 2006 Workshop in Computational Aesthetics: Artificial Intelligence Approaches to Happiness and Beauty*, páginas 61–69. 2006.

Haslum, P. Narrative planning: Compilations to classical planning. *Journal of Artificial Intelligence Research*, vol. 44, páginas 383–395, 2012.

Hayes-Roth, B. A blackboard architecture for control. *Artificial intelligence*, vol. 26(3), páginas 251–321, 1985.

Helbig, H. *Knowledge representation and the semantics of natural language*. Springer, 2006.

Hertz, J., Krogh, A. y Palmer, R. Introduction to the theory of neural computation. 1990.

Kartal, B., Koenig, J. y Guy, S. J. User-driven narrative variation in large story domains using monte carlo tree search. En *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, páginas 69–76. 2014.

Kazman, R., Klein, M. y Clements, P. Evaluating software architectures for real-time systems. *Annals of Software Engineering*, vol. 7(1-4), páginas 71–93, 1999.

Klein, S. Automatic novel writer: A status report. *Papers in text analysis and text description*, 1973.

Kolodner, J. *Case-based reasoning*. Morgan Kaufmann, 2014.

Lakoff, G. Structural complexity in fairy tales. 1972.

Lebowitz, M. Creating characters in a story-telling universe. *Poetics*, vol. 13(3), páginas 171–194, 1984.

Lebowitz, M. Story-telling as planning and learning. *Poetics*, vol. 14(6), páginas 483–502, 1985a.

Lebowitz, M. Storytelling and generalization. En *Seventh Annual Conference of the Cognitive Science Society*, páginas 100–109. 1985b.

Lem, S. *Ciberíada*. Alianza Editorial, 2021.

León, C. y Gervás, P. Creativity in Story Generation From the Ground Up: Non-deterministic Simulation driven by Narrative. En *5th International Conference on Computational Creativity, ICCC 2014*. Ljubljana, Slovenia, 2014.

León, C. y Gervás, P. Creativity in story generation from the ground up: non-deterministic simulation driven by narrative. En *5th International Conference on Computational Creativity, ICCC*. 2014.

Li, B., Lee-Urban, S., Johnston, G. y Riedl, M. Story generation with crowd-sourced plot graphs. En *AAAI*. 2013.

Li, B., Lee-Urban, S. y Riedl, M. O. Toward autonomous crowd-powered creation of interactive narratives. En *5th Workshop on Intelligent Narrative Technologies, Palo Alto, CA*, vol. 8, páginas 25–52. 2012.

Li, B. y Riedl, M. O. Scheherazade: Crowd-powered interactive narrative generation. En *AAAI*, páginas 4305–4306. 2015.

Liu, H. y Singh, P. Makebelieve: Using commonsense knowledge to generate stories. En *AAAI/IAAI* (editado por R. Dechter y R. S. Sutton), páginas 957–958. AAAI Press / The MIT Press, 2002.

Liu, H. y Singh, P. Conceptnet—a practical commonsense reasoning tool-kit. *BT technology journal*, vol. 22(4), páginas 211–226, 2004.

Malkewitz, S. G. R. y Iurgel, I. Technologies for interactive digital storytelling and entertainment. En *TIDSE*. Springer, 2006.

Maranda, P. Semiography and artificial intelligence. *International Semiotic Spectrum*, vol. 4, páginas 1–3, 1985.

Margolin, U., Hühn, P., Meister, J. C., Pier, J. y Schmid, W. The living handbook of narratology. 2013.

Marsella, S. C., Pynadath, D. V. y Read, S. J. Psychsim: Agent-based modeling of social interactions and influence. En *Proceedings of the international conference on cognitive modeling*, vol. 36, páginas 243–248. Citeseer, 2004.

McIntyre, N. y Lapata, M. Plot induction and evolutionary search for story generation. En *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, páginas 1562–1572. 2010.

Meehan, J. R. Tale-spin, an interactive program that writes stories. En *In Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, páginas 91–98. 1977.

Méndez, G., Gervás, P. y León, C. A model of character affinity for agent-based story generation. En *9th International Conference on Knowledge, Information and Creativity Support Systems, Limassol, Cyprus*, vol. 11, página 2014. 2014.

Méndez, G., Gervás, P. y León, C. On the use of character affinities for story plot generation. En *Knowledge, Information and Creativity Support Systems*, páginas 211–225. Springer, 2016.

Menéndez, R. *Cinco golpes de genio*. Alba Editorial, Barcelona, Spain, 2013.

Minsky, M. A framework for representing knowledge. in book: Winston ph, hom b, eds. the psychology of computer vision. 1975.

Minsky, M. *Society of mind*. Simon and Schuster, 1988.

MongoDB, I. Mongodb official site. `https://www.mongodb.com/`, 2017. [Online; accessed 29-December-2017].

Montfort, N., Pérez, R., Harrell, D. F. y Campana, A. Slant: A blackboard system to generate plot, figuration, and narrative discourse aspects of stories. En *Proceedings of the fourth international conference on computational creativity*, páginas 168–175. 2013.

Morse, H. Preliminary operating notes for saga ii. Informe técnico, MIT Technical Memorandum 8436-M-29, 1960.

Newman, S. *Building microservices: designing fine-grained systems*. .ºReilly Media, Inc.", 2015.

Oinonen, K., Theune, M., Nijholt, A. y Uijlings, J. Designing a story database for use in automatic story generation. En *International Conference on Entertainment Computing*, páginas 298–301. Springer, 2006.

Ong, T. y Leggett, J. J. A genetic algorithm approach to interactive narrative generation. En *Proceedings of the fifteenth ACM conference on Hypertext and hypermedia*, páginas 181–182. 2004.

Papazoglou, M. P. Service-oriented computing: Concepts, characteristics and directions. En *Web Information Systems Engineering, 2003. WISE 2003. Proceedings of the Fourth International Conference on*, páginas 3–12. IEEE, 2003.

Peinado, F. y Gervás, P. Minstrel reloaded: from the magic of lisp to the formal semantics of owl. En *International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, páginas 93–97. Springer, 2006.

Pemberton, L. *Story structure:a narrative grammar of nine chansons de geste of the Guillaume dÓrange cycle*. Tesis Doctoral, University of Toronto, 1986.

Pemberton, L. A modular approach to story generation. En *Proceedings of the fourth conference on European chapter of the Association for Computational Linguistics*, páginas 217–224. Association for Computational Linguistics, 1989.

Perez y Perez, R. *MEXICA: A Computer Model of Creativity in Writing*. Tesis Doctoral, The University of Sussex, 1999.

Porteous, J., Charles, F. y Cavazza, M. Plan-based narrative generation with coordinated subplots. En *European Conference on Artificial Intelligence (ECAI 2016)*, vol. 285, páginas 846–854. IOS Press, 2016.

Propp, V. Morphology of the folktale. 1928. 1968.

Randell, B. y Buxton, J. Software engineering techniques: Report of a conference sponsored by the nato science committee, rome, italy, 27th-31st october 1969. 1970.

Riedl, M. O. y Young, R. M. Narrative planning: balancing plot and character. *Journal of Artificial Intelligence Research*, vol. 39(1), páginas 217–268, 2010.

Rozanski, N. y Woods, E. *Software systems architecture: working with stakeholders using viewpoints and perspectives*. Addison-Wesley, 2011.

Rumelhart, D. Notes on a schema for stories. En *Representation and Understanding: Studies in Cognitive Science* (editado por D. G. Bobrow y A. Collins), páginas 211–236. Academic Press, Inc, New York, 1975a.

Rumelhart, D. E. Notes on a schema for stories. *Representation and understanding: Studies in cognitive science*, vol. 211(236), página 45, 1975b.

Rusell, S. y Norvig, P. Artificial intelligence: A modern approach. *Pretice Hall Series in Artificial Intelligence*, vol. 1, páginas 649–789, 2003.

Ryan, J. Grimes' fairy tales: a 1960s story generator. En *International Conference on Interactive Digital Storytelling*, páginas 89–103. Springer, 2017.

Schank, R. C. The primitive acts of conceptual dependency. En *Proceedings of the 1975 workshop on Theoretical issues in natural language processing*, páginas 34–37. Association for Computational Linguistics, 1975.

Schank, R. C. y Abelson, R. P. *Scripts, plans, and knowledge*. Yale University New Haven, CT, 1975a.

Schank, R. C. y Abelson, R. P. Scripts, plans, and knowledge. En *Proceedings of the 4th International Joint Conference on Artificial Intelligence - Volume 1*, IJCAI'75, páginas 151–157. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1975b.

Schank, R. C. y Abelson, R. P. Scripts, plans, goals, and understanding. *Lawrence Erlbaum: Hillsdale, New Jersey*, 1977.

Schank, R. C. y Abelson, R. P. *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*. Psychology Press, 2013.

Schank, R. C. y Lebowitz, M. *Does a hippie own a hairdrier?*. Yale University, Department of Computer Science, 1979.

Sharples, M. *How We Write: Writing As Creative Design*. Routledge, 1999.

Shaw, M. y Clements, P. The golden age of software architecture. *IEEE software*, vol. 23(2), páginas 31–39, 2006.

Shaw, M. y Garlan, D. *Software architecture: perspectives on an emerging discipline*. Prentice-Hall, Inc., 1996.

Si, M. *Thespian: a decision-theoretic framework for interactive narratives*. Tesis Doctoral, University of Southern California, 2010.

Si, M., Marsella, S. C. y Pynadath, D. V. Thespian: Modeling socially normative behavior in a decision-theoretic framework. En *Intelligent Virtual Agents*, páginas 369–382. Springer, 2006.

Singh, P., Lin, T., Mueller, E. T., Lim, G., Perkins, T. y Zhu, W. L. Open mind common sense: Knowledge acquisition from the general public. En *OTM Confederated International Conferences."On the Move to Meaningful Internet Systems"*, páginas 1223–1237. Springer, 2002a.

Singh, P., Lin, T., Mueller, E. T., Lim, G., Perkins, T. y Zhu, W. L. Open mind common sense: Knowledge acquisition from the general public. En *On the move to meaningful internet systems 2002: Coopis, DOA and Odbase*, páginas 1223–1237. Springer, 2002b.

Smith, T. C. y Witten, I. H. A planning mechanism for generating story text. *Literary and Linguistic Computing*, vol. 6(2), páginas 119–126, 1991.

Stephens, R. *Beginning software engineering*. John Wiley & Sons, 2015.

Swartjes, I. *The plot thickens: bringing structure and meaning into automated story generation*. Proyecto Fin de Carrera, EEMCS: Electrical Engineering, Mathematics and Computer Science, 2006.

Tearse, B., Mawhorter, P., Mateas, M. y Wardrip-Fruin, N. Skald: minstrel reconstructed. *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6(2), páginas 156–165, 2014.

Thorndyke, P. W. Cognitive structures in comprehension and memory of narrative discourse. *Cognitive psychology*, vol. 9(1), páginas 77–110, 1977a.

Thorndyke, P. W. Cognitive structures in comprehension and memory of narrative discourse. *Cognitive Psychology*, vol. 9, páginas 77–110, 1977b.

Tolstoy, L. *War and peace*. Courier Dover Publications, 2017.

Turner, S. R. *Minstrel: A Computer Model of Creativity and Storytelling*. Tesis Doctoral, University of California at Los Angeles, Los Angeles, CA, USA, 1993. UMI Order no. GAX93-19933.

Van Harmelen, F., Lifschitz, V. y Porter, B. *Handbook of knowledge representation*. Elsevier, 2008.

Vernet, J., editor. *The Arabian Nights*. Editorial Planeta, 1990.

Vukotic, A., Watt, N., Abedrabbo, T., Fox, D. y Partner, J. *Neo4j in action*. Manning Publications Co., 2014.

Wardrip-Fruin, N. *Expressive Processing: Digital fictions, computer games, and software studies*. MIT press, 2009.

Ware, S. G. y Young, R. M. Cpocl: A narrative planner supporting conflict. En *AIIDE*. 2011.

Wilensky, R. Story grammars versus story points. *Behavioral and Brain Sciences*, vol. 6(4), páginas 579–591, 1983.

Winston, P. H. The strong story hypothesis and the directed perception hypothesis. En *2011 AAAI Fall Symposium Series*. 2011.

Winston, P. H. The genesis story understanding and story telling system a 21st century step toward artificial intelligence. Informe técnico, Center for Brains, Minds and Machines (CBMM), 2016.

Wojcik, R., Bachmann, F., Bass, L., Clements, P., Merson, P., Nord, R. y Wood, B. Attribute-driven design (add), version 2.0. Informe técnico, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst, 2006.

Wolff, E. *Microservices: Flexible Software Architecture*. Addison-Wesley Professional, 2016.

Yar, M., Rahmati, V. y Oskouei, H. A survey on evolutionary computation: Methods and their applications in engineering. *Modern Applied Science*, vol. 10, página 131, 2019.

Yazdani, M. *Generating events in a fictional world of stories*. Department of Computer Science, University of Exeter, 1983.

Yngve, V. H. *Random generation of English sentences*. Massachusetts Inst. of Technology, 1961.

Zhu, J. y Ontañón, S. Shall i compare thee to another story?—an empirical study of analogy-based story generation. *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 6(2), páginas 216–227, 2013.

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes