Story Plot Generation based on CBR

Pablo Gervás, Belén Díaz-Agudo, Federico Peinado, Raquel Hervás
Dep. Sistemas Informáticos y Programación, Universidad Complutense de Madrid
Madrid, Spain
{pgervas,belend}@sip.ucm.es,fpeinado@fdi.ucm.es,

Abstract

In this paper we present a system for automatic story generation that reuses existing stories to produce a new story that matches a given user query. The plot structure is obtained by a case-based reasoning (CBR) process over a case base of tales and an ontology of explicitly declared relevant knowledge. The resulting story is generated as a sketch of a plot described in natural language by means of natural language generation (NLG) techniques.

1 Introduction

With the advent of information technology, many domains have achieved productivity improvements due to the introduction of computers. The film industry has seen some of its tasks radically affected – like the generation of special visual effects, or the computer assisted animation. On related work areas, such as the generation of film scripts, there has been no comparable improvement. A number of films in industry have benefited from IT tools for script-writing, such as and Dramatica[1] or WritePro[2], but there has been little progress towards automation of the process of draft generation.

Automatic construction of story plots has always been a long standing goal in the entertainment industry, specially in the more commercial genres that are fuelled by a large number of story plots with only a medium threshold on plot quality, such as TV series or video games. Although few professionals would contemplate full automation of the creative processes involved in plot writing, many would certainly welcome a fast prototyping tool that could produce a large number of acceptable plots involving a given set of initial circumstances or restrictions on the kind of characters that should be involved. Subsequent selection and revision of these plot sketches by professional screen writers could produce revised, fully human-authored valid plots.

In this paper we present a system for automatic story generation that reuses a case base of existing stories to produce a new story that matches a given user query. The plot structure is obtained by a case-based reasoning (CBR) process over the case base and an ontology of explicitly declared relevant knowledge (Section 3). The resulting story is generated as a sketch of a plot described in natural language by means of natural language generation (NLG) techniques (Section 4).

[1] http://www.screenplay.com/products/dpro/index.html
[2] http://www.writepro.com/

2 Related Work

This section, outlines previous relevant work on story generation and natural language generation applied to narrative texts.

2.1 Computational Plot Generation

There have been various attempts in the literature to obtain a computational model of story generation by considering the role of planning in story generation. Tale-Spin [15], IDA (I-Drama Architecture) [13] and Fairclough and Cunningham [9] explore with different approaches the way in which plot generation can be interpreted as a planning process.

Of the various approaches to plot generation, some have applied CBR explicitly to the plot generation process. Minstrel [19] included a CBR process to model the creative process of generating a story line. Fairclough and Cunningham [9] developed an interactive multiplayer story engine that applies case-based planning and constraint satisfaction to control the characters and make them follow a coherent plot. In [4] poetry generation is chosen as an example of the use of the COLIBRI system. COLIBRI assists during the design of KI-CBR systems that combine cases with various knowledge types and reasoning methods. It is based on CBROnto [6, 7], an ontology that incorporates reusable CBR knowledge, including terminology plus a library of reusable Problem Solving Methods (PSMs).

Our work is based on ideas from the work of Vladimir Propp [17]. Propp derives a morphological method of classifying tales about magic, based on the arrangements of "functions". This provides a description of the folk tales according to their constituent parts, the relationships between those parts, and the relations of those parts with the whole. The main idea is that folk tales are made up of ingredients that change from one tale to another, and ingredients that do not change. According to Propp, what changes are the names - and certain attributes - of the characters, whereas their actions remain the same. These actions that act as constants in the morphology of folk tales he defines as *functions*. For example, some Propp functions are: Villainy, Departure, Interdiction, Interdiction Violated, Acquisition of a Magical Agent, Guidance, Testing of the hero, etc. There are some restrictions on the choice of functions that one can use in a given folk tale, given by implicit dependencies between functions: for instance, to be able to apply the *Interdiction Violated* function, the hero must have received an order (*Interdiction* function).

There have been various attempts to apply this work to story generation. The GEIST project [10] uses a simplified version of Propp's morphology. Fairclough and Cunningham's [9] interactive multiplayer story engine operates over a way of describing stories based on Propp's work.

2.2 Natural Language Generation

The most natural format for presenting a plot to users is narrate it in natural language. Obtaining a high quality natural language text for a story is itself a subject of research even if the plot is taken as given [3]. This paper is considered

a first approximation to the overall task, and it focuses on the interface between the CBR module and the NLG module.

Recent overviews of NLG [18] define a number of basic tasks to be carried out when generating a natural language text: *content determination* - finding what to say -, *document structuring* - organizing what is to be said -, *sentence aggregation* - grouping together the parts that allow it -, lexicalization - selecting the words that will realize each concept -, *referring expression generation* - choosing the right expression to refer to each element in its actual context - and *surface realization* - turning the result into a linear natural language sentence. Different solutions can be applied to resolve each of this tasks.

Text generation in this paper is done by means of NLG based on templates. Texts follow conventionalized patterns which can be encapsulated in *schemas* [14], template programs which produce text plans. Schemas are derived from a target text corpus by breaking them up into messages, organising the messages into a taxonomy, and identifying how each type of message is computed from the input data.

The specific architecture of the NLG module presented here is implemented using cFROGS [1], a framework-like library of architectural classes intended to facilitate the development of NLG applications. cFROGS identifies three basic design decisions: what set of modules to use, how control should flow between them, and what data structures are used to communicate between the modules.

3 A CBR System for Composing Plot Plans

The plot generation module of the system presented here is based on a Knowledge Intensive CBR approach to the problem of generating story plots from a set of cases consisting of analyzed and annotated fairy tales. Our system operates in two phases: an initial one that applies CBR to obtain a plot plan from the conceptual description of the desired story provided by the user, and a final phase that transforms the resulting plot plan into a textual rendition by means of template based NLG. Readers interested specifically in the plot generation process are referred to [5], where this initial stage is described in detail. This section provides only a brief outline of those parts of the plot generation process that are relevant to the NLG module.

3.1 The System Knowledge

Knowledge representation in our system is based on a basic ontology which holds the various concepts that are relevant to story generation. Propp's character functions are used as basic recurrent units of a plot. In order to be able to use them computationally, they have been translated into this ontology that gives semantic coherence and structure to our cases [16]. This initial ontology is subject to later extensions, and no claim is made with respect to its ability to cover all the concepts that may be necessary for our endeavour.

We have implemented this ontology using the last release of the Protégé ontology editor that was developed at Stanford University [8]. It can manage

ontologies in OWL [2], a new standard that has recently reached a high relevance. The choice of OWL as a representation language provides the additional advantage, that it is designed to work with inference engines like RACER [11].

Although the functions of the *dramatis personae* are the basic components, we also have other elements. The ontology also provides the background knowledge required by the system, as well as the respective information about characters, places and objects of our world.

The ontology is used to complement the specific knowledge of the cases, and it is used by the CBR system to measure the semantical distance between similar cases or situations, and maintaining an independent story structure from the simulated world. The domain knowledge of our application is the classic fairy tale world with magicians, witches, princesses, etc. The current version of the ontology contains a number of basic subconcepts to cover this additional domain knowledge that needs to be referred from within the represented function. Next we summarize the most important concepts of our ontology.

- **Propp functions**. Propp's character functions act as high level elements that coordinate the structure of discourse. Each function has constraints that a character that is to perform it must satisfy. The contents of a function are the answers to the Wh-questions: what, when, where, who (the characters of the function) and why.

- **Moves**. Morphologically, a tale is a whole that may be composed of *moves*. A move is a type of development proceeding inside the history, and it can refer different Propp's functions. One tale may be composed of several moves that are related between them. We represent tales and their composing moves using structured descriptions. A tale is related with an ordered sequence of complete moves. We represent the temporal sequence between these moves using the CBROnto temporal relations.

- **Character**. The roles in the story must be filled by characters (for instance, who performs a function). Each character is defined by a set of relationships with other characters, objects in his possession, location... These characters are one of the elements that the user can choose to customize a story.

- **Properties of the characters** By properties or attributes of the characters, we mean the totality of all the external qualities of the characters: their age, sex, status, external appearance, peculiarities of this appearance, dwelling (described by a relation with a place),... These attributes provide the tale with its brilliance, charm and beauty. However, one character in a tale is easily replaced by another (permutability law) [17].

- **Roles**. Propp describes a number of 'spheres of action' that act as roles that certain characters have to fulfill in the story (like villain, hero,..).

- **Places and objects**. Certain locations (outdoors, indoors, countries, cities...) and symbolic objects (towels, rings, coins...) can be significant to

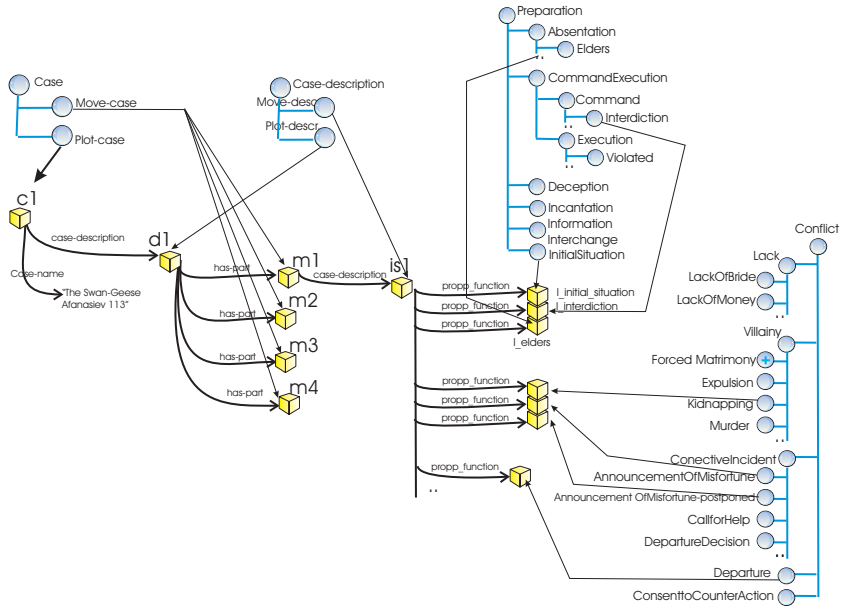

Figure 1: Case Structure

the way a story develops, and any sort of substitution during adaptation must take this into account. Our ontology must have the ability to classify such locations and objects.

- **Descriptions**. Since our system generates text by filling in templates with selected descriptions that correspond to instances of particular concepts, it was considered convenient to have these descriptions represented in the ontology in such a way that their relations with the relevant concepts can also be modelled and the inference mechanisms available can be employed in their selection.

- **Cases**. Cases are built up of a selection of stories from the original set of the Afanasiev compilation originally used by Propp. Within the defined case structures we represent the plots of the fairy tales. Cases are described with attributes and values that are pre-defined, and structured in an object-oriented manner (see Figure 1).

We facilitate the case structure authoring tasks by proposing a framework to represent cases that is based on the CBROnto terminology. Besides, we define a reasoning system (based on generic CBR PSMs) that works with such representations [7].

Cases are built based on CBROnto case representation structure [6, 7] using the vocabulary from the domain ontology. The semantic constraints between scene transitions are loosely based on the ordering and co-occurrence constraints established between Proppian functions.

As an example of the type of stories that are being considered, the following outline of one of the tales that Propp analyses is given below [3]. The main events of the plot are described in terms of character functions (in bold) :

> *The Swan Geese (113 of Afanasiev Collection).* **Initial situation** (a girl and her small brother). **Interdiction** (not to go outside), **interdiction violated**, **kidnapping** (swan geese take the boy to Babayaga's lair), **Competition** (girl faces Babayaga), **Victory**, **Release from captivity**, **Test of hero** (swan geese pursue the children), **Sustained ordeal** (children evade swan geese), **Return**.

3.2 The CBR Module

We use the descriptive representation of the tale plots with a CBR system, that retrieves and adapts these plots in several steps and using the restrictions given in the query.

3.2.1 Query specification, similarity assessment and case retrieval

We propose an interactive case retrieval process taking into account progressive user inputs. A query determines some of the components of the tale we want to build. For example, its characters, descriptive attributes, roles, places, and the Propp functions describing the actions involved in the tale. The system retrieves the more similar case with the restriction of Propp morphology and characters available. As CBROnto provides with a general test-bed of CBR methods we have made different tests with different similarity measures between the complex descriptions that represents the plots [7].

Each retrieved case constitutes a plot-unit template. The retrieved case components are *hot-spots* (or flex points) that will be substituted with additional information obtained from the context, i.e. the query, the ontology and other cases, during the adaptation process. Similarity measures should guarantee that (when possible) all the query elements are valid elements to be allocated in the retrieved cases.

For instance, let us say we want a story about a *princess*, where **murder** occurs, where an **interdiction** is given and **violated**, there is a **competition**, and a **test of the hero**. We can use that information to shape our query. The system retrieves the case story number 113, Swan-Geese (described earlier).

Retrieval has occurred using structural similarity over the ontology (described in [7]) because the structure of this story satisfies straight away part of the conditions (interdiction, competition, test of hero) imposed by the query. No murder appears, but there is a *similar* element: a kidnapping. **Kidnapping** and **murder** are similar because they are different types of villainies; so, they are represented as children of the same concept **Villainy** in the ontology (see Figure 2).

[3] Complete text in http://gaia.sip.ucm.es/people/fpeinado/swan-geese.html

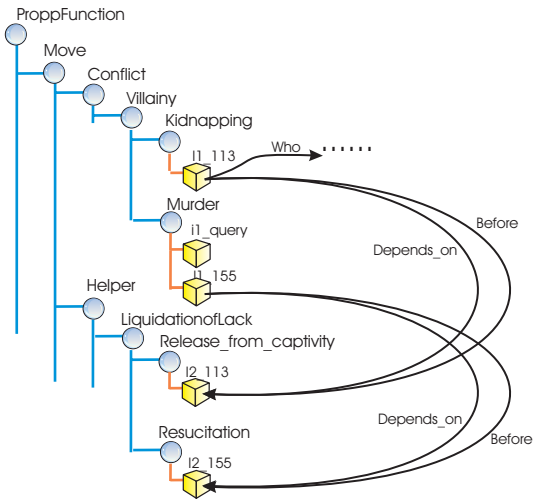

Figure 2: Substitution example

3.2.2 Adaptation

The retrieval process provides with the plot skeleton where the system makes certain substitutions. More creative results may be achieved by generating a solution as a mixture of the ingredients from various cases. During the adaptation of our *plot case*, we use additional retrieval steps (defining adequate queries) over the case base of *move cases* (that are part of the plot cases, see Figure 1) to find appropriate substitutes maintaining the dependencies and temporal relations.

In our example, the system should suggest an adaptation where **murder** is substituted for the **kidnapping**. However, the **kidnapping** in the retrieved case has *dependencies* with the **release from captivity** that appears later on (which is a **liquidation of lack** according to the ontology) (see Figure 2). To carry out a valid adaptation, the adaptation process is forced to define a query and retrieve cases in which **murder** appears with a *similar* dependency (i.e. dependency with another **liquidation of lack**).

The following case is retrieved (only a part of which is relevant to the issue):

> (*155 of Afanasiev Collection*). (...) **Absentation** of the hero (brother goes hunting), **Deception** of the villain (beautiful girl entices him), **Murder** (girl turns into lioness and devours him), (...) **Consent to counteraction** (other brother sets out), **Competition** (faces beautiful girl), **Victory** (kills lioness), **Resurrection** (revives brother), **Return**.

In this case there is a dependency between the **murder** and the **resuscitation**. The adaptation system can therefore substitute the kidnapping-release pair in the first retrieved case with the murder-resuscitation pair in the second,

cf0 **Initial situation** **(a knight and his beloved princess).**	what: ∅ where: l0 who: (ch0,ch1)
cf1 **Interdiction** **(not to go forest).**	what: action(ch2,ch1,s0,warn), (clause(s0,rep,neg), action(ch1,x,l1,go)) where: l0 who: (ch2,ch1)
cf2 **Interdiction violated**	what: action(ch1,l1,go) where: (l0,l1) who: (ch1)
cf3 **Murder** **(a lioness devours her)**	what: action(ch3,ch1,devour) where: l1 who: (ch3,ch1)
cf4 **Competition** **(knight faces the lioness)**	what: action(ch0,ch3,face) where: l1 who: (ch0,ch3)
cf5 **Victory** **(kills lioness)**	what: action(ch0,ch3,kill) where: l1 who: (ch0,ch3)
cf6 **Resurrection** **(revives the princess)**	what: action(ch0,ch1,resurrect) where: l1 who: (ch0,ch1)
cf7 **Return**	what: action(ch0,l0,return) where: (l1,l0) who: ch0

Table 1: Plot plan for *The Lioness (new fairy tale).*

obtaining a better solution for the given query. Additional adaptations can be carried out to substitute the hero of the first case (the girl) or the prisoner (the boy) for the princess specified in the query.

The resulting plot, showing how the two cases are combined, could be a story like the one shown in Table 1. Details are discussed in the next section.

4 The Natural Language Generation Module

Processing the plot plan in order to obtain a readable rendition of it involves operations like joining sentences together wherever this gives more fluency to the text, substituting character identifiers for pronouns or definite descriptions wherever appropriate, or selecting specific words to use in realizing templates whenever options are available. The NLG involved in this process is simple, based on templates rather than a full grammar. However, it provides a marked improvement in the readability of resulting texts.

4.1 Input Data for the NLG Module

If the CBR process of the first stage has taken place successfully, the second stage will accept as input a data structure satisfying the following constraints:

- The case that has been selected during retrieval, has been pruned or combined with other cases retrieved during adaptation and strung into a case sequence that makes up a plot skeleton.

- The character functions, acting as templates for the basic units of the plot, have been filled in during adaptation with identifiers for the characters described in the query

We will refer to it as a *plot plan*. A plot plan is structured in terms of instances of character functions. Each instance of a character function contains references to specific instances of other concepts in the ontology. To avoid ambiguity problems during processing, each instance of a concept in the ontology must have a unique identifier. Table 1 presents an example of the plot plan for the result obtained by the CBR module for the example discussed earlier. For readability, wh-questions that are not directly relevant to the plot have been ommitted. Cross indexing with instances in the ontology is indicated by an identifier (`cf*` for character functions, `l*` for locations, `ch*` for characters...).

The amount of information available in the ontology for the various concepts now involved in the plot plan is very large. Additionally, rendering such a story into text must take into account contextual dependencies that arise progressively as subsequent character functions are described in sentences. For these reasons, transcription of the plot plan into text is broken down into small fragments of text corresponding to single character functions.

Each character function is processed separately. When processing a given character function, the NLG module accesses only the information available in the ontology corresponding to:

1. The instance of the character function that appears in the text
2. The instances of all elements of the ontology that appear in the instantiation of the character function
3. The instances of all elements of the ontology that appear in the instantiations in 2.

Let us take the `cf3` instance of the *Murder* character function appearing in Table 1 as an example. The following information about it, provided by the CBR module by navigating the ontology according to the above rule, is available to the NLG module:

- The events related with this character function take place at some location: an instance of *location* that is a dark forest outside the castle
- Two *characters* are involved: a lioness and the princess
- The lioness has the following *atributes*: she plays the *role* of villain in the story, she is hungry and fierce, and she lives in the forest
- The princess has the following *atributes*: she plays the role of *victim* in the story, she is blonde and pretty, has parents, is in love with a knightt, and lives in the castle

All this information is processed by the *Conceptual Representation* stage of the cFROGS architecture, and the required data are converted into an internal

Table 2: Output of the stages of the NLG module

Content Determination	Discourse Planning	Sentence Aggregation
character(ch1,princess) character(ch3,lioness) location(l1,forest) role(ch3,villain) attribute(ch3,hungry) attribute(ch3,fierce) action(ch3,ch1,devour)	character(ch3,lioness), attribute(ch3,hungry) character(ch3,lioness), attribute(ch3,fierce) character(ch3,lioness), character(ch1,princess), action(ch3,ch1,devour)	character(ch3,lioness), attribute(ch3,hungry), attribute(ch3,fierce) character(ch3,lioness), character(ch1,princess), action(ch3,ch1,devour)

Ref. Expression Gen.	Lexicalization	Surface Realization
character(ch3,lioness), ref(ch3,def), attribute(ch3,hungry), attribute(ch3,fierce)	"lioness" "a" "hungry" "fierce" L(x)+" was "+L(y)+" and "+L(z)	"The lioness was hungry and fierce."
character(ch3,pron), character(ch1,princess), ref(ch1,def), action(ch3,ch1,devour)	"she" "princess" "the" L(x)+" devoured "+L(y)	"She devoured the princess.".

representation `Draft` data structure chosen by the designers. This data structure presents the advantage of allowing simultaneous representation at multiple levels – for instance semantic representations, documental representations, and rhetorical representations.

Some text has been generated before this character function, which acts as the context in which it will appear in the final story. To deal with this, the NLG module also keeps a record of the results of processing all previous character functions of the same plot. This is stored as an additional `Draft` data structure, containing the *discourse history*. This discourse history is initially empty, and it is stored by the NLG module and updated each time a new character function is processed.

4.2 Stages of the NLG Module

The NLG module carries out its task in six stages: content determination, discourse planning, sentence aggregation, referring expression generation, lexicalization, and surface realization. Each one of these stages corresponds to a basic task in rendering the instance of the character function as text. An example of the output at each stage for our example is presented in Table 2.

4.2.1 Content Determination

This stage checks the information that is available for the given instance of the character function against a record of the semantic information that has already been rendered as text for previous character functions of the same plot plan. Its main goal is to filter out of the rendition of the present character

function the information that has already been conveyed in the rendition of the previous ones.

Because the discourse history is represented as a `Draft` data structure, the system has access to the semantic information that acted as input in previous character functions, even though they have already been processed into text.

For the example character function described above, a lot of the information available will have already been imparted in the transcription of earlier character functions, and so is filtered out of the contents to be processed for the current one. For instance, most of the information about the princess will already be known to the reader at this stage and is ommitted. The lioness, however, enters the story at this stage, and so will have to be described. The location is not new either, since the princess moved there in the previous character function, so its description will be ommitted.

4.2.2 Discourse Planning

The discourse planning stage has to organise all the information that has been selected as relevant into a linear sequence, establishing the order in which it is to appear in the final text.

This is achieved by means of simple heuristics that give priority in the sequence to some concepts of the ontology over others. For instance, whenever they are all to be included in a text rendition, characters are treated first, then location, then any relatives of the character are mentioned, then objects present are mentioned, then finally the actions in the character function are narrated. For the example above, the available information is reordered according to the heuristic.

4.2.3 Sentence Aggregation

This stage takes care of regrouping any elements in the current draft that may be aggregated to provide better fluency of the text. The algorithm employed looks for concepts appearing as object of similar verb constructions that occur adjacently in the discourse plan, and substituting them for a single verb construction with a plural object.

In the example, the system spots a possible rhetorical grouping of two of the entries, joining the descriptive facts about the lioness as a conjunction.

4.2.4 Referring Expression Generation

This stage checks the current version of the draft against the lexical information already used in the rendition of any previous character functions of the same plot plan. The goal is to ensure that each instance of a concept appearing in the current character function is referred to in terms that: (1) are sufficient for the reader to identify with the current one any previous appearances of the same concept, and (2) are not exactly the same as those used in the immediately preceding sentence, to improve the fluency of the text.

Few alternatives are available for a system handling limited world information such as ours. The current version of this stage focuses on substitution

of nouns for pronouns whenever the name has recently been mentioned in the text. The window of occurrence that is contemplated is restricted to a character function unless there is a possibility of confussion.

An additional task handled at this stage is selection of definite or indefinite articles to accompany singular nouns. Following Heim [12], a simple mechanism is implemented that opts for indefinite articles whenever the referrent has not appeared before in the story and definite articles everywhere else.

For the example, a definite article is used for the lioness in her first appearance, because she has already been mentioned earlier. The following reference to her uses a pronoun. The princess is given a definite article because she has already appeared (not a pronoun, since earlier occurrences where in other character functions).

4.2.5 Lexicalization

This stage selects words to describe the concepts involved in the current draft. Depending on the different syntactic roles that concepts are associated to, the actual terms to be used can be of three types:

- Concepts that are static object (characters, locations, objects...) are associated to specific lexical terms
- Concepts that involve verbs (whether introduced explicitly by instances of character functions or implicitly required to provide linguistic structure to descriptions of available static objects) are associated with templates corresponding to the structure of the sentences to be built with them
- Concepts related to linguistic features (definite/indefinite markers, pronouns...) are given the correct form

Templates partly solve the need for having an explicit grammar, but the ontology provides the required information to solve issues like number and gender agreement.

The example presents the type of lexical elements identified for the instances under consideration.

4.2.6 Surface Realization

This stage is in charge of using the terms selected in the previous stage to fill in the templates. Additionally, it carries out a basic ortographic transformation of the resulting sentences. Templates are converted into strings formatted in accordance to the orthographic rules of English - sentence initial letters are capitalized, and a period is added at the end.

5 Conclusions

Although the system is not fully-implemented yet, the progress so far points to a reasonable solution for Story Generation. Our approach follows the lines of structuralist story generation, which distinguishes it from more transformational work [15, 13]. Unlike the uses of Proppian functions described for other

systems [9], our approach represents character functions with more granularity. This allows the establishment of relations between characters and attributes and the functions in which they appear. Dependencies between character functions are modelled explicitly, so they can be checked and enforced during the process of plot generation without forcing the generated plots to be structurally equivalent to the retrieved cases.

The heuristics currently used for the NLG module are intended as tentative approximations, and further work is planned in which the results of the system are evaluated for readability by means of reading tests by a group of volunteer users. The heuristics would then be refined to achieve maximal user satisfaction.

The system architecture is general in as much as one accepts Propp's set of character functions as complete. In the face of disagreement, the ontology is easy to extend, and, as mentioned before, it is not intended to be complete as it is. Under these conditions, the approach described in this paper may be extend to work in other domains.

In future work we intend to address the possible interactions between the two stages. Once these issues have been solved, the integration of the generator with software applications for script writing – along the lines of classics - such as Dramatica and WritePro- can be contemplated.

Acknowledgements

The work was partially funded by the Spanish Committee of Science & Technology (TIC2002-01961). The third author is supported by a FPI Predoctoral Grant from Complutense University of Madrid.

References

[1] R. H. and P. Gervás. Using design patterns to abstract a software architecture for natural language generation. In S. Juknath and E. Jul, editors, *ECOP 2004 PhD workshop*, 2004.

[2] S. Bechhofer and F. van Harmelen et al. OWL web ontology language reference. W3C http://www.w3.org/TR/2004/REC-owl-ref-20040210/, 2004.

[3] C. B. Callaway and J. C. Lester. Narrative prose generation. *Artificial Intelligence*, 139(2):213–252, 2002.

[4] B. Díaz-Agudo, P. Gervás, and P. González-Calero. Poetry generation in COLIBRI. In S. Craw and A. Preece, editors, *ECCBR 2002, Advances in Case Based Reasoning*. Springer LNAI, 2002.

[5] B. Díaz-Agudo, P. Gervás, and F. Peinado. A case based reasoning approach to story plot generation. In *ECCBR'04*, Springer-Verlag LNCS/LNAI, Madrid, Spain, 2004.

[6] B. Díaz-Agudo and P. A. González-Calero. An architecture for knowledge intensive CBR systems. In *Advances in Case-Based Reasoning – (EWCBR 2000)*. Springer-Verlag, 2000.

[7] B. Díaz-Agudo and P. A. González-Calero. A declarative similarity framework for knowledge intensive CBR. In *Procs. of the (ICCBR 2001)*. Springer-Verlag, 2001.

[8] J. G. et al. The evolution of Protégé: An environment for knowledge-based systems development. Technical report, Stanford University, 2002.

[9] C. Fairclough and P. Cunningham. A multiplayer case based story engine. In *4th International Conference on Intelligent Games and Simulation*, pages 41–46. EUROSIS, 2003.

[10] D. Grasbon and N. Braun. A morphological approach to interactive storytelling. In M. Fleischmann and W. Strauss, editors, *Artificial Intelligence and Interactive Entertainment, Living in Mixed Realities*, 2001.

[11] V. Haarslev and R. Moller. *RACER User s Guide and Reference Manual Version 1.7.7*. Concordia University and Univ. of Appl. Sciences in Wedel, November 2003.

[12] I. Heim. *The Semantics of Definite and Indefinite Noun Phrases*. PhD thesis, University of Massachusetts, 1982.

[13] B. Magerko. A proposal for an interactive drama architecture. In *AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, Stanford, CA, 2002. AAAI Press.

[14] K. R. McKeown. The text system for natural language generation: An overview. In *20th Annual Meeting of the ACL*, pages 261–265, 1982.

[15] J. R. Meehan. Tale-spin and micro tale-spin. In R. C. Schank and C. K. Riesbeck, editors, *Inside computer understanding*. Erlbaum Lawrence Erlbaum Associates, Hillsdale, NJ, 1981.

[16] F. Peinado, P. Gervás, and B. Díaz-Agudo. A description logic ontology for fairy tale generation. In *Language Resources for Linguistic Creativity Workshop, 4th LREC Conference.*, Lisboa, Portugal, 2004.

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

[18] E. Reiter and R. Dale. *Building Natural Language Generation Systems*. 2000.

[19] S. R. Turner. Minstrel: A computer model of creativity and storytelling. Technical Report UCLA-AI-92-04, Computer Science Department, 1992.