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.

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

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

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 definition 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 im-

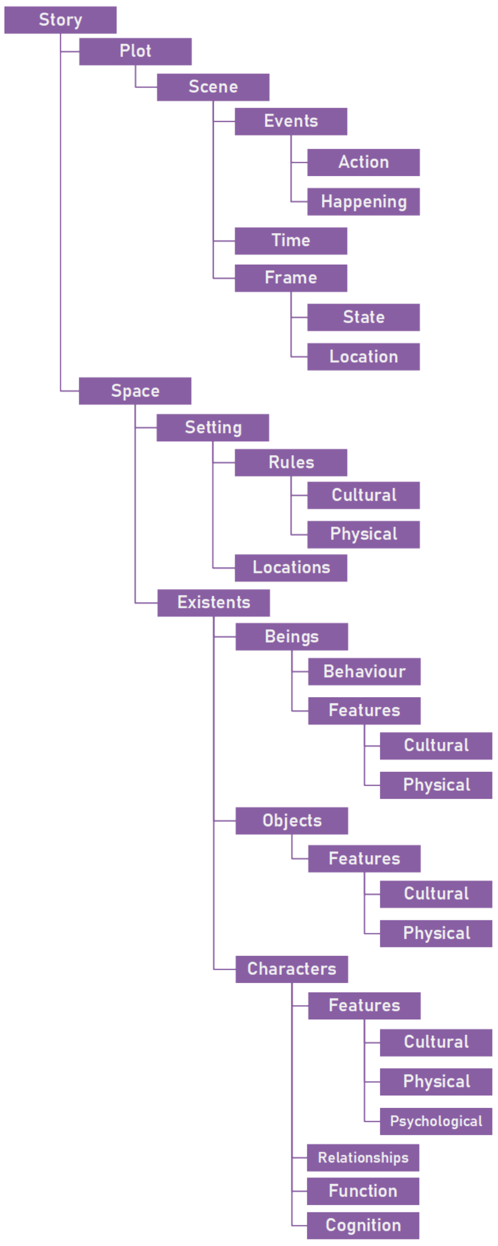

Figure 1. Stories common representation model.

pact 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.

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.

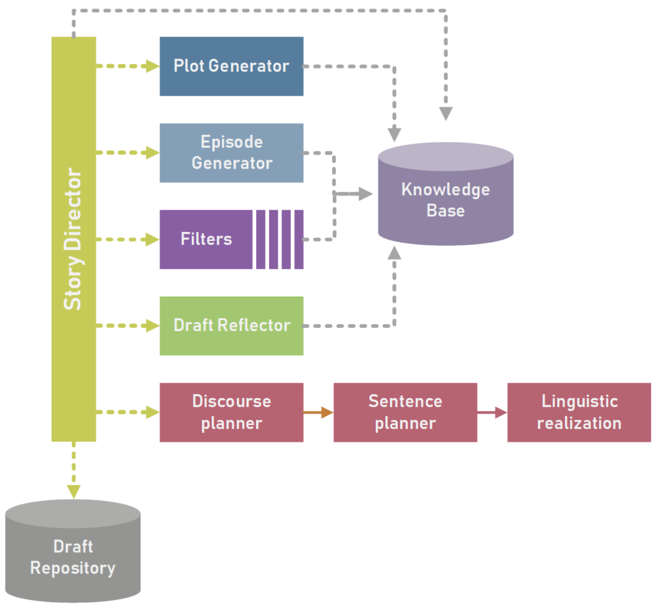

Figure 2. Architecture of Afanasyev.

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

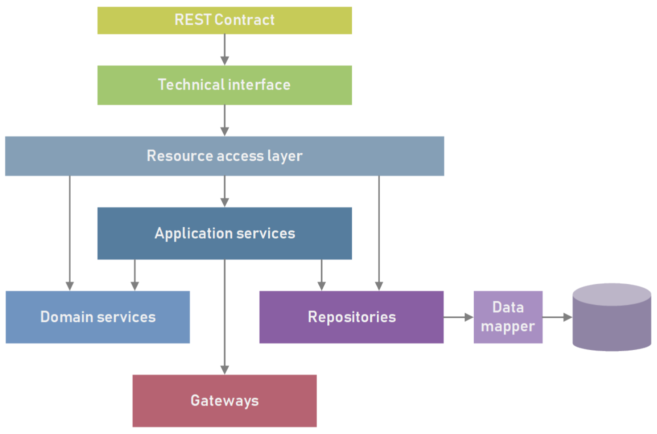

Figure 3. Story director architecture.

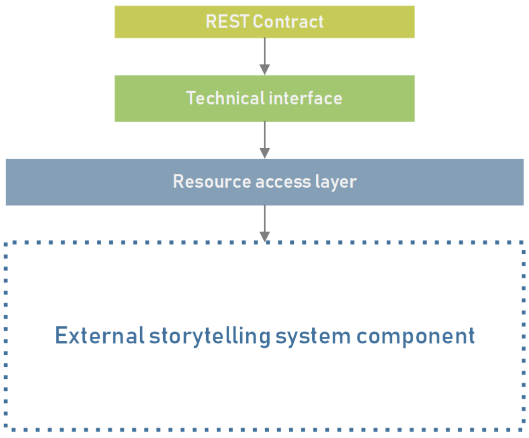

Figure 4. Marker microservices architecture.

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)

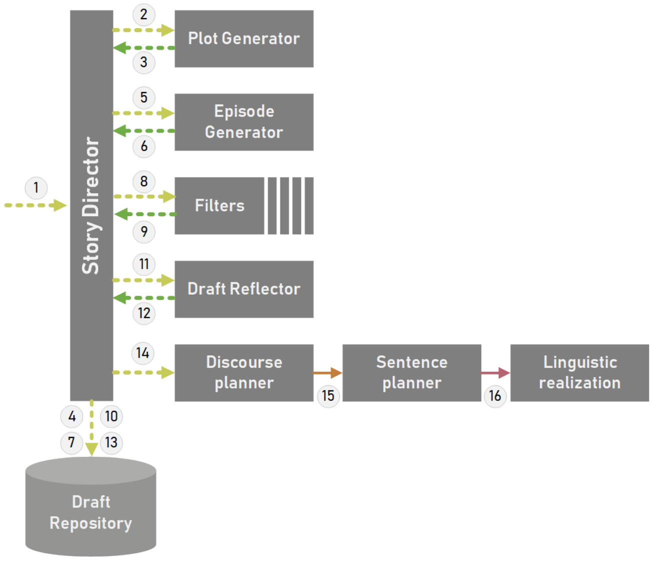

Figure 5. Operation of Afanasyev.

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).