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).

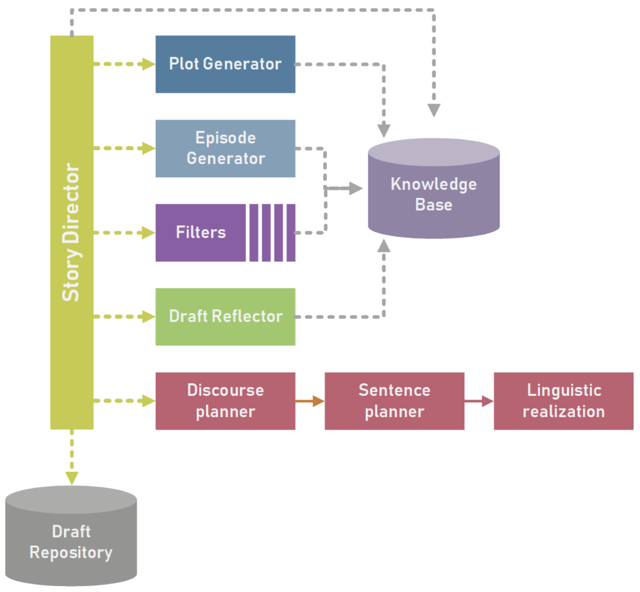

Figure 1: Architecture of Afanasyev.

- 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

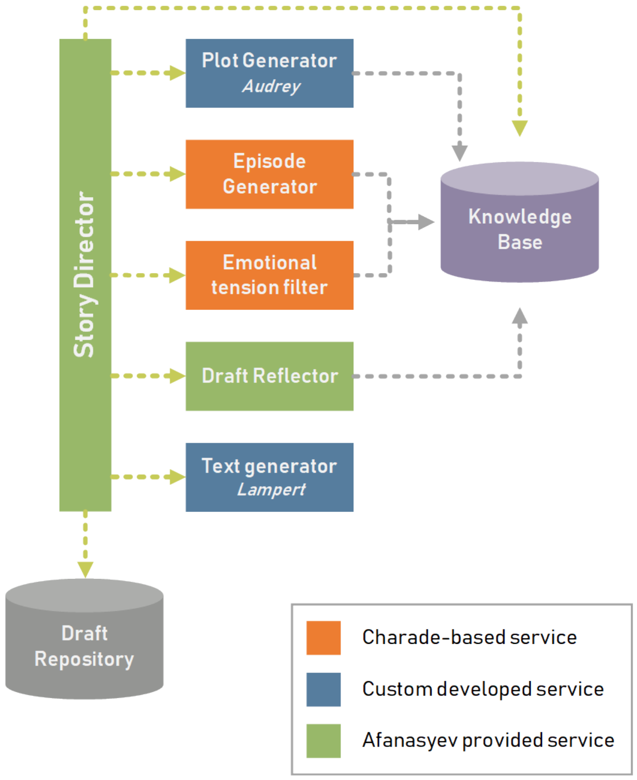

Figure 2: Architecture of INES.

shared set of concepts each time a new system joins the ecosystem. Hence, every contributor performs an initial load expressing its rules.

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

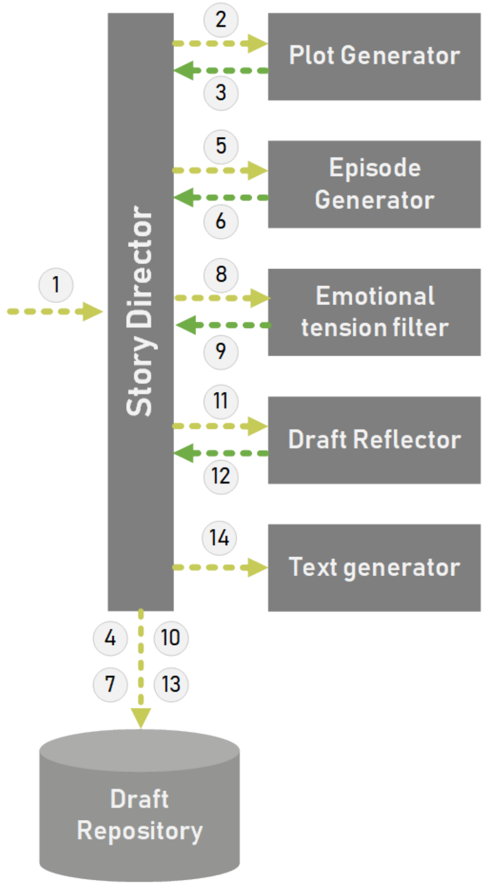

Figure 3: Operation of INES / Afanasyev.

- 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

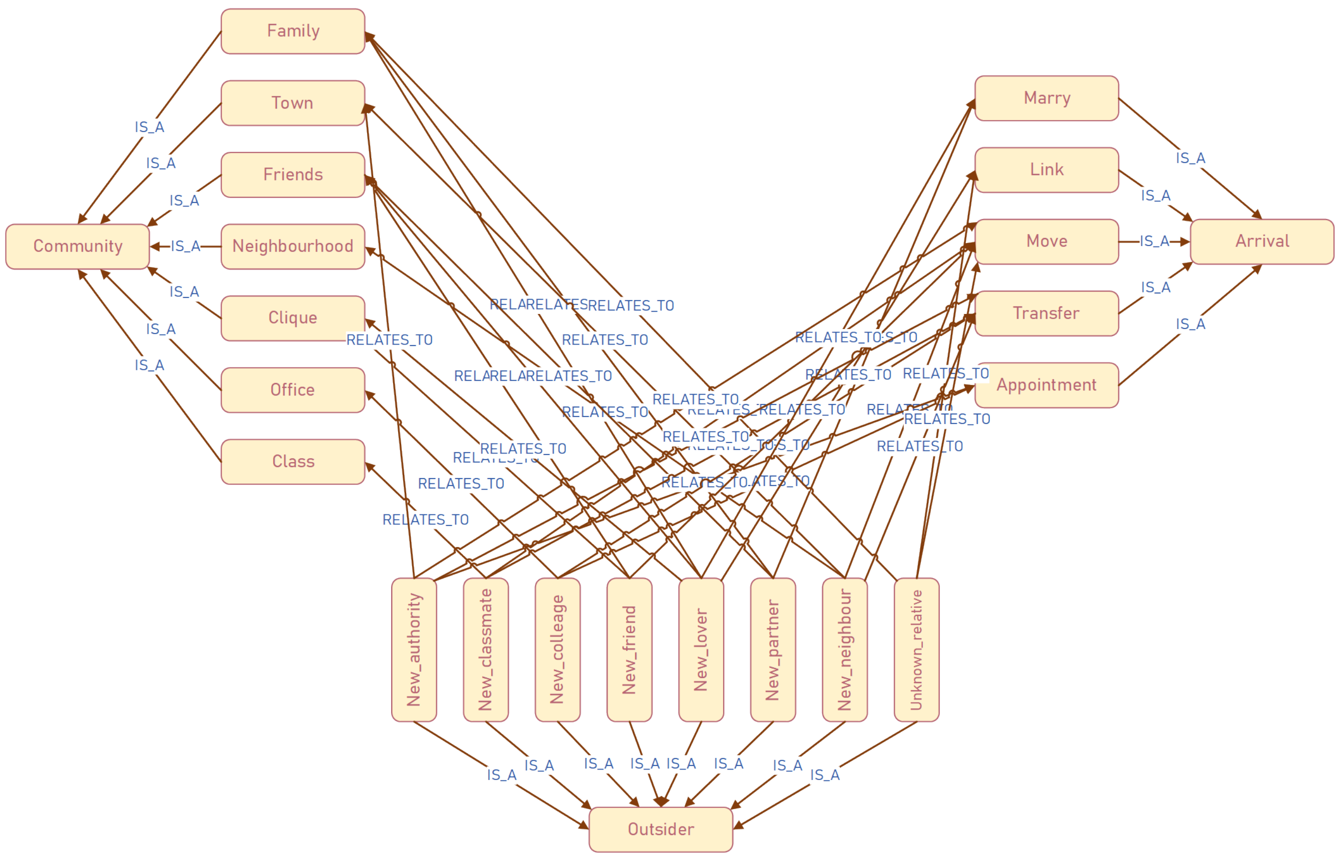

Figure 4: Partial view of the concepts relationships in the KB.

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.