Mike Sennott

Hamilton College

Narrative Generation Survey

1. Introduction

Narrative generation is the field of automating the process of authoring a narrative. In simpler terms, it describes any program or system designed to construct and output a story. The majority of early narrative generation systems created prose narratives, but more recent studies have focused on extending narrative generation towards a dynamic and often interactive storytelling process.

The overall goal of narrative generation is to develop a system that can approach human storytelling facilities. Such a system would be an important step towards computational creativity and intelligence. In order to dialogue naturally with humans, computerized systems and artificially intelligent agents should be able to exercise the synthesis and abstraction inherent in story generation [Smith and Witten, 1991]. In theory, any system undergoing the Turing Test should be able to tell a narrative indistinguishable from those extemporaneously invented by humans.

Though the definition of computational creativity, much like the definition of artificial intelligence, is a matter of much debate, Ritchie's three criteria for the creativity of a computationally generated artifice, novelty, quality, and typicality [Ritchie, 2007], provide a good framework for qualitatively evaluating the value of an artificial narrative. When applied to stories, novelty describes how original the work is within the genre or set of stories that served to inspire it; quality describes the logical consistency, narrative interestingness, and quality of writing; and typicality describes how recognizable the work is as a well-structured story. Many papers on narrative generation describe design philosophies strikingly similar to these criteria. While they are debatable as proscriptive guidelines, the criteria are useful for comparing the relative successful of various narrative generation

system, and this paper will utilize them in that capacity.

Though the primary benefit of narrative generation is progress towards achieving theoretical definitions of artificial intelligence and creativity, the field has practical applications in entertainment and education. A system with the ability to generate interesting and novel stories would have obvious effects on the entertainment industry. While such systems would never supplant human writers entirely, popular media such as television and magazines would likely benefit from an inexpensive and reliable story generator. However, given the current state of narrative generation, the odds are against the development of any such system in the near future. Interactive story generation has more immediate applications. Computer games can integrate narrative generation to introduce variety and replay value, benefits which lead Szilas to state that the field, which he calls Interactive Drama, represents the "ultimate challenge of digital entertainment" [Szilas, 2005]. Interactive systems with dynamically generated stories have a wide range of applications in education, from interactive children's stories to occupational training simulators [Prada et al., 2000] [Marsella and Gratch, 2002].

The genesis of narrative generation took place in the late 1970s, when two competing approaches arose: procedural and declarative systems. In procedural systems, the interaction of character agents generates emergent narratives. Declarative systems derive a narrative from the abstract concept of a story by using rules based on story structure. More recent systems have attempted to bridge the gap between the two approaches, combining the beneficial aspects of both types of system.

2. Procedural Systems

2.1 Basic Procedural Model

Procedural systems seek to generate narratives through the planning process of AI agents that represent characters within the environment of the story. The first such system was called TALE-SPIN

[Meehan, 1978]. Its approach became the basis for procedural narrative generation, and many subsequent systems have offered improvements and variations on its formula.

The character agent must have one or more goals that it will seek to satisfy during the story. TALE-SPIN uses bodily needs such as hunger and shelter as these primitive goals. The story generation process runs a planning algorithm on the goals of a main character to determine which actions the character will take to attempt to achieve those goals. The planning algorithm in TALE-SPIN is modeled after planning systems such as STRIPS [Fikes and Nilsson, 1971]. It decomposes primitive goals into transitive subgoals, using a predefined set of action plan templates, or planboxes, to decide on a sequence of actions. Each planbox has a set of logical preconditions and postconditions, describing the conditions under which a character can execute the action sequence contained in the planbox and the consequences of executing those actions. A character agent in TALE-SPIN is able to prioritize plans, recognizing when they have failed and when changing environmental knowledge indicates that they should be retried.

TALE-SPIN includes character personalities, relationships, and physical mapping, all of which are implemented as part of the pre- and postconditions of the planning process. Characters have variable degrees of kindness, honesty, vanity, and intelligence, and cannot enact certain actions unless they have a proper value for one or more of those attributes. For example, a precondition of lying is being dishonest. Character relationships work in a similar manner. Multiple factors go into defining the relationship each character agent holds with each other character. Each parameter appears as a precondition in certain actions, and other actions modify these parameters as postconditions. For example, a character will not ask a favor of another character he or she hates, and if one character gives a gift to another, the latter will feel affection towards the former. Finally, TALE-SPIN keeps track of physical locations. Relative position appears as the precondition for certain actions, and changing location is the postcondition of movement. Each character agent contains a model representing its knowledge of the physical environment. TALE-SPIN implements this model as a hierarchy of maps

and submaps, including more detail in submaps which the agent has visited to represent gaining knowledge about an area.

To summarize, the basic procedural narrative generation model, as first defined by TALE-SPIN, contains a model of a physical environment and various character agents that populate it. Each character agent has goals, personality traits, interpersonal relationships, and knowledge of the physical environment. A central planning process integrates all of this information, deciding upon a sequence of character actions. TALE-SPIN outputs a simple text representation of these actions to form the narrative, outputting personality and relationship traits as they are relevant to the planning process. Appendix A contains an example narrative generated by TALE-SPIN.

2.2 Improvements on the Procedural Model

Subsequent procedural narrative generation systems use the same basic model as TALE-SPIN, but with improvements to the world model, character agent, or planning process. One early system, TAILOR, uses a different planning process from TALE-SPIN. Whereas TALE-SPIN is essentially a negotiation-based agent framework, TAILOR seeks to introduce conflict into the narrative by treating the story as a competitive game [Smith and Witten, 1991]. There are two only character agents, representing a protagonist and an antagonist. The story plans each characters' actions by running an adversarial search. The two character agents evaluate states of the story world using inverse heuristics, meaning that the antagonist's goals directly oppose the protagonist's goals. There is a one-to-one correspondence between character actions, moves in the game, and sentences in the output story text. In other words, each agent in turn evaluates the move that would minimize or maximize the heuristic through adversarial search, then TAILOR applies the effects of the corresponding action to the world model and outputs a sentence corresponding to that action. The protagonist can fail to achieve its goals if the antagonist “wins.” Appendix A contains an example narrative generated by TAILOR.

DAYDREAMER is at heart the implementation of a model for generating daydreams [Mueller

and Dyer, 1985], essentially short fantastic narratives in which planning focuses on a single character (the dreamer). DAYDREAMER introduces a basic emotional model to procedural narrative generation. In the DAYDREAMER model, events modify the emotional state of the agent. Certain emotions trigger "control goals," representing common daydream themes such as revenge or rationalization, which the agent then composes and enacts action plans to fulfill. As opposed to TALE-SPIN, in which characters plan according to a single primary need, DAYDREAMER keeps a weighted hierarchy of disparate, potentially conflicting goals, prioritizing according to the emotional model. Another important innovation of DAYDREAMER is a dynamic memory, which the planner can use to apply a plan from memory to an analogous current situation instead of repeating the planning process.

The Oz project is a multifaceted system towards "developing technology for dramatic, interactive, simulated worlds" [Bates et al., 1992]. It uses a more self-contained and comprehensive character agent structure than TALE-SPIN, called "Tok." Because Oz is meant to facilitate narratives through interactive environments instead of prose, its agents must be more robust than those in TALE-SPIN. The basic character agent model in TALE-SPIN makes discrete and static decisions, but since a user within the Oz environment can enter input at any time, Tok must be able to both react to continuous input and handle the environment changing during planning. As such, Tok uses a sense-think-act cycle, integrating a sensory model, a model for reactivity and goal-directed behavior ("Hap"), and an emotional and social model ("Em"). Tok uses systems for language analysis and generation, called "Gump" and "Glinda," respectively, to interact with the user through text input and output.

A relatively recent system integrates a more sophisticated emotional model than those found in Tok or DAYDREAMER into a procedural interactive system [Marsella and Gratch, 2002]. In this model, as in previous models, a character agent develops a plan based on its emotional state and beliefs about the environment. However, unlike previous models, emotions can modify an agent's beliefs – if the agent encounters a negative emotion, it can "cope" by either deciding on an action plan or revising its own beliefs. If the agent revises beliefs, it will reappraise the environment and previous events in

the context of the new beliefs, leading to a new set of emotions. Many other theories of cognitive modeling in artificially intelligent agents would be directly applicable to procedural narrative generation through use as character agents.

MAKEBELIEVE is a narrative generation system that can generate a great variety of stories [Liu and Singh, 2002]. It differs primarily from previous procedural systems in that it draws actions from a large database of "common sense" rather than any set of actions defined specifically for the narrative situation. The user starts off the story with a sentence, from which MAKEBELIEVE derives a cause and effect. Then, using fuzzy logic based off semantic networks, it searches each possible consequent action for validity, matching similarity of subject, manner, and cause of the candidate with effect of the precedent. MAKEBELIEVE generates a short narrative based on the resulting chains of causality. Though these narratives are generally unsatisfying as stories, the implementation of a "common sense" database is an important step towards comprehensive planning facilities. Appendix A contains an example narrative generated by MAKEBELIEVE.

IDtension is a procedural system designed with theoretically minimal narrative structures [Szilas and Rety, 2004]. The IDtension system contains characters, goals, values, tasks (actions towards goals), and obstacles (hindering tasks). Characters appear as nodes in a graph, with tasks as edges that connecting characters to goals nodes. Obstacles are conditions that must be met in order to "pass" through a task to the desired effect, and goals can have effects that satisfy these conditions. In other words, obstacles are preconditions of actions, and goals have postconditions, creating a hierarchy of subgoals. Mathematical positions in the graph are meaningful, as the axes of the graph are ethical values. A character's ethical position will determine the tasks they use to achieve a goal, allowing for simple stories to have a moral message.

2.3 Advantages and Disadvantages of the Procedural Model

The main advantage of the procedural model is that it is easily adaptable to interactive

storytelling, as seen in some of the aforementioned systems [Bates et al., 1992] [Marsella and Gratch, 2002]. As long as character agents are able to react to input at any time, the user can simply replace the reasoning facilities of a character agent, receiving some sensory or textual input and deciding on actions. Character agents can be made more robust for an interactive system by utilizing a subgoal hierarchy, in other words, having multiple ways of solving any given goal. That way, if unexpected user input disrupts an agent's plan, it can adapt by switching to an alternate plan [Cavazza et al., 2002].

Another advantage of procedural narrative generation is the wide variety of narratives that such systems can generate. Emergent stories can often play out in various unexpected ways, bounded only by the permutations of potential goals and actions. With a large enough set of goals and actions, as in MAKEBELIEVE [Liu and Singh, 2002], or with the unpredictability that a user character introduces in an interactive system, a procedural system has a good chance of generating a story with some aspect of novelty. Procedural narratives are also generally guaranteed to be logical and consistent. The system derives events directly from logical action sequences and character motivations, so actions cannot seem nonsensical or “out of character” if their pre- and postconditions are sound.

The primary disadvantage of procedural narrative generation systems is that they do not guarantee that a narrative will be dramatic – a narrative could theoretically be entirely devoid of conflict. A conventional story follows the pattern of a beginning establishing the setting and characters, then rising action building tension towards a climax, then the ending resolving tension, but procedural narratives will not necessarily adhere to that structure. Between those two factors, a procedurally generated story could be unrecognizable as a story, bearing much more similarity to a simple narration of mundane events. In other words, there is no guarantee about the typicality of a procedurally generated story. Additionally, since a good story should be interesting, the quality of a story suffers at low levels of narrative tension, which the basic procedural narrative generation model has no way of preventing.

2.4 Lessening the Disadvantages of the Procedural Model

In summary, the main disadvantage of procedural narrative generation is that it does not guarantee an interesting story. However, certain procedural systems have introduced improvements that allow for the production of more consistently interesting and story-like narratives.

UNIVERSE is one example, a narrative generation system that seeks to make interesting stories by making interesting and detailed characters for storytelling [Lebowitz, 1983]. Like TALE-SPIN, each character has a set of values representing character traits, and another representing the parameters of its relationship with each other character. In addition, UNIVERSE can apply preset "historical" events that have occurred prior to the beginning of the narrative and affect characters' relationships and goals. UNIVERSE non-deterministically instantiates the personalities, relationships, histories, and goals of various characters within the story environment. The goals and history events in UNIVERSE are based on those in soap operas to maximize tensions and drama. Theoretically, by using a set of complex and interesting characters, a procedural narrative generation program should be more likely to generate a complex and interesting story.

A more recent example is Thespian [Si et al., 2005]. Like TALE-SPIN, Thespian is a procedural narrative generator in which the user sets the system into motion by defining the initial character goals and personalities. The character agent is more complex than that in TALE-SPIN, featuring complex memory, a belief model, and a system of multiple weighted goals. Thespian's main innovation is the ability for the user to enter "preferred" scripts, or desirable stories from the set of those that the initial environment can generate. A fitting algorithm automatically alters character personalities and goal weights to make the "preferred" outcome more likely. The user can review the changes made by the fitting algorithm, and choose whether to repeat the process or run the generator. The entry of preferred scripts essentially trains the system to recognize what the user considers a good story, making it more likely that it will generate a story that corresponds to the user's subjective ideas of story quality.

Fabulist seeks to make procedural narratives more interesting and creative by making them less predictable [Riedl and Young, 2005]. It is a procedural narrative generation system using the basic model of running a planning algorithm on character goals. Fabulist's primary innovation is Initial State Revision, in which certain non-crucial details of the initial world are left indeterminate. If the preconditions of an action require that an indeterminate value be true or false, the system can revise that value accordingly, "filling in" details about the initial world state. Fabulist also considers character traits and certain action preconditions as "recommendations" rather than "requirements," allowing them to be subverted in an unexpected way under certain circumstances. Both the Initial State Revision and requirements system allow for generated stories to be less dependent on the constraints of the initial world model, and thus less predictable and more novel.

3. Declarative Systems

3.1 Story Grammars

Whereas a procedural narrative generation system works from the bottom up, generating emergent stories based on the plans of character agents, declarative systems work from the top down. The most straightforward type of declarative narrative generation system is a story grammar. A story grammar system uses grammatical rules to break the abstract concept of a story down into its components, eventually decomposing it into concrete events and the text of the story. In other words, a story grammar system builds a tree from the top down, in which the start node is an abstract story and the terminal nodes represent the narrative text. The first story grammar narrative generation system, TELLTALE [Correira, 1980], uses rules based on extended Horn clauses. Whereas standard rules expand some pattern of story "macrostructures" into a sequence of additional macrostructures, the rules in TELLTALE contain additional logical pre- and postconditions. In this way, TELLTALE isolates the grammatical expansion of the story from the logical causes and effects of actions within the story.

GESTER implements a more inclusive and detailed story grammar system [Pemberton, 1989]. The GESTER grammar starts with a "complex story," which breaks down into "simple stories" combined by causality, motive, chronologically, or by having the same actor. The simple story breaks down into precondition, event, and postcondition, and the event breaks down into motivation, plan, qualification, action, and resolution. In each event, the acting character plans to acquire the knowledge and means to perform the action, in steps that are usually withheld from the text of the story. GESTER introduces a layer of abstraction between the actual representation of events, or "storyline," and how they are told, as they can differ in chronology, for example. As such, the terminal elements of the grammar are divided into tied motifs, which serve to relate the story line, and free motifs, which serve to make the narrative richer and more story-like through descriptions. Appendix A contains an example narrative generated by GESTER.

The Joseph system adds two interesting improvements to the basic story grammar model. The first is placing terminals such as objects, characters, and events into equivalence classes [Beaubouef and Lang, 1998]. Such groupings evenly distribute chances that a certain type of terminal will be chosen rather than just a certain terminal. For example, if either a harmful or helpful object can appear at a certain point in a story, the odds of which appears should be independent of how many different helpful or harmful objects are present in the story. In other words, this allows for adding terminals without skewing the chance of selecting a certain type. Joseph also uses an alternate precondition system to guarantee the logical consistency of generated stories, called explanation closure [Stewart and Lang, 1998]. Once the system derives a terminal state from the story grammar tree, it will search back up the tree to find the node representing the cause of the state, proceeding to recursively find the cause of each such node until reaching the initial world state. Appendix A contains an example narrative generated by Joseph.

TWISTER uses rules and story grammars to make a story with a twist ending [Platts et al., 2002]. It takes in an existing story as input, then uses lexical analysis to divide it into discrete events,

also identifying the climax and main character. TWISTER uses TELLTALE-like "scripts," or events with pre- and postconditions, to work backwards and determine possible motivations that the characters may have had for performing actions. Then, TWISTER determines "hidden" motivations and events that do not contradict the given text – in other words, alternate rules that fulfill the preconditions derived from the given narrative. Finally, TWISTER works forward in the story grammar tree to find the results of the "hidden" events and motivations, generating a set of events. The final narrative places the generated events, which should reveal the twist, after the climax of the input narrative, resulting in a story with a twist,

3.2 Advantages and Disadvantages of Declarative Narrative Generation

Declarative narrative generation has nearly the opposite advantages and disadvantages of the procedural model. Where procedural systems do not guarantee conflict, story-like structure, or any other element that separates "stories" from "narratives," narratives derived from story grammars will have all of these, because they are the direct decompositions of rules corresponding to the conventions of story structure. Moreover, if the story grammar includes rules tailored to the conventions of a certain genre, as in TAILOR [Pemberton, 1989], the narrative will likewise inherit the conventions of that genre. This guarantees typicality, as the narratives will be recognizable both as stories and members of a certain genre. In addition, storytelling conventions should imply a certain baseline of story quality, if only due to the presence of some conflict and a recognizable climax.

However, because declarative narratives come directly from story conventions, they will be strictly conventional stories, likely bearing strong similarities to the stories upon which the system was based. Additionally, since declarative systems build narratives from the top down, they lack the procedural model's aspect of emergent unpredictability as to how the system's elements will interact and combine. Thus declarative systems as a whole are far less likely to produce a novel story than procedural systems.

Finally, declarative systems are far more difficult to adapt to interactivity than procedural systems. At each potential decision that the user can make, a story branches into multiple possible narratives, each of which the grammar must be able to explicitly define. A top-down system in which the user makes frequent decisions, never mind continuous interaction, will therefore have an extraordinarily high branching factor, making truly interactive declarative narratives intractable [Szilas, 2005].

3.3 Adapting Declarative Narrative Generation to Interactivity

Despite the intractability of "true" interactive declarative narrative generation, there are some ways of allowing the user some interactivity while circumventing the need to branch. Many interactive declarative narrative generation systems use a method based on Vladimir Propp's morphology of folklore. In this morphology, Propp divides the common folk tale into set scenes, each of which contains a certain type of event [Propp, 1968]. The characters, settings, and specifics may vary, but folktales will almost always follow the same basic sequence of scenes. For example, a scene in which the hero marries his or her love will always take place near the end of the story. This morphology provides a good framework for computational generating narratives: a declarative system can determine the characters, settings, and events that comprise each scene, then simply proceed to narrate them according to the morphological sequence. The user has some ability to affect the proceedings of the narrative within each scene, but the following scene will always be one predetermined by the morphology. A morphological system can introduce limited branching by allowing multiple possible scenes to follow the current one by depending on the user's actions. In other words, the user may interact freely with the narrative on a shallow and immediate level, but interaction at the macroscopic level, affecting the overall progression of the narrative, is simple branching constrained by predefined scene sequences.

Story-engine is a morphological narrative generator integrated with an authoring environment

[Schneider et al., 2003]. Within the authoring environment, the user enters the parameters of each scene, including its role in the Propp morphology, its setting, and various "scene-properties" that determine how the engine will represent the scene. Then, Story-engine tells the story interactively using three-dimensional graphics. Story-engine is part of the Geist project, which gives users a virtual tour of German history. M. is a similar system to Story-engine, with the added feature of using natural language parsing to infer the Propp function of a scene from a prose description [Hartmann et al., 2005]. M. uses simple reflexive character agents to provide robust reactions to the user's actions within each scene, and automatically transitions to another scene based on certain stimulus, such as the passage of time or the user's actions.

Facade uses a similar combination of reactive character agents and branching scenes to create interactivity [Mateas and Stern, 2000]. However, it is not based on the Propp morphology, instead telling variations of a single short story in which the user visits an unhappy married couple for dinner. Facade uses a combination of graphics and text for storytelling: the characters and environment appear as three-dimensional graphics to the user, and the user interacts with the reactive character agents through text-based conversation. Each scene in Facade is comprised of a precondition, a postcondition, and a set of "beat" character actions that sequence together to construct the scene. In the sense of having both pre- and postconditions and a decomposition pattern, a scene in Facade is analogous to an extended Horn-clause rule in TELLTALE. A Facade narrative is comprised of a sequence of scenes, each chosen at runtime based on fulfillment of the preconditions and a postcondition that works towards the predetermined narrative arc of the story. In other words, Facade contains a model of the general plot of the narrative, then strings together scenes to approximate that plot. Each scene is a series of beats, or interactions between the player and characters. In a single beat, the characters say or do something and the user responds through text input. Each beat can change certain values in the story model, such as the extent to which the couple's marital problems have been revealed or how much one of the couple likes the user's character. Since these values are used as the preconditions for scene

selection, the user's responses to beats have an effect on the macroscopic plot progression, in a process similar to branching. The character agents in Facade are based on the Hap model from the Oz project. The agents take advantage of Hap's advanced language generation and analysis systems, but do not use any of the Hap's internal planning facilities for anything other than sequencing proper animations. Instead, the agent in Facade includes alterations that allow for the reception of a set of commands during each beat [Mateas and Stern, 2002]. In summary, the declarative story in Facade decomposes from the overall plot to scenes to beats. The user can affect which scenes appear in the story and which beats appear in each scene. However, Facade can only generate a very small number of macroscopic plots, and the overall story arc is coded into the system instead of generated by the system.

4. Intermediate Systems

4.1 Sequential Hybrids

Several different narrative generation systems fall neither into the category of strict procedural nor declarative generation. Instead, they combine aspects of both character agent planning and derivation of story structure. The most simple intermediate systems simply run the two separately in sequence, combining the results to make a cohesive narrative.

BRUTUS is one system that combines many elements of previous systems [Bringsjord and Ferrucci, 1999]. First, it determines a theme, such as betrayal or self-deception, to use in initializing the goals and personalities of character agents. Then, it runs a procedural system, much like the one found in TALE-SPIN, to generate the actual events of the story. Those events are passed to a declarative story grammar, which uses rules of story structure to generates the actual text of the story, incorporating elements from GESTER such as rearranging the order of events and adding descriptions. As Pérez y Pérez and Sharples state, “BRUTUS’ contribution consists of merging different known methodologies into one program.” [Pérez y Pérez and Sharples, 2003]. Appendix A contains an

example of a narrative produced by BRUTUS.

LOGTELL is an intermediate narrative generation system based on the Propp morphology. It is significantly different from the morphological examples discussed in Section 3.3 for three reasons. First, user interaction occurs during the actual generation of the plot, as opposed to the dramatization [Karlsson et al., 2006]. In other words, the user takes the place of the author, not the reader. Secondly, character roles, not scenes or events, are determined by Propp roles – in other words, the story will feature characters representing a hero, victim, villain, etc. Third, because the morphology is character-based, a procedural system determines narrative events. LOGTELL follows a three-step process: goal inference, plan generation, and user interference. In goal inference, LOGTELL analyzes the configuration of the world state to determine goals for each character role. For example, if the victim is vulnerable, the villain might plan to kidnap him/her. In plan generation, the system uses procedural character planning to determine action plans for each character agent to achieve its goal. In user interference, the user can demand that the previous goal inference and plan generation steps be repeated to generate a new plot. If not, the story up to that point is dramatized graphically. Essentially, LOGTELL is a procedural narrative generated that uses a combination of archetypal character roles and user feedback to produce a story that is interesting and well-structured.

4.2 Reader Modeling

One interesting outlier among narrative generation systems is the reader model designed by Paul Bailey [Bailey, 1999]. Rather than building a story based on grammars or character goals, it develops the story by imagining a reader's response. A logical ontology represents the world model of the story environment. The algorithm generates every possible "segment," or event, imaginable by the reader model. Using a ontological hierarchy, new segments can come from generalizing, specializing, combining, or removing sentences in the world model ontology. After each segment of a story, an algorithm develops expectations and questions that readers might have. Expectations come from

following logical premises and rules of inference, and can have varying strengths. Questions can come from details that that were omitted from the previous segment, or previous expectations that were defied by the previous segment. The system chooses segments corresponding to a standard story structure. the beginning should be straightforward, then rising action should gradually build tension by raising questions and defying expectations, then the conclusion should provide resolutions by answering questions.

4.3 Author Modeling Via Pattern-Matching

Whereas reader modeling is unique, author modeling has become a popular narrative generation schema. Author modeling systems are based on a cognitive model of how human authors go about writing creative stories. The first renown author modeling system was MINSTREL [Turner, 1993]. It is centered around a case-based problem solver and episodic memory, but involves elements of both procedural and declarative narrative generation systems. At its upper levels, MINSTREL generates a story out of author schema and character schema. Author schema contain goals and corresponding action plans for scenes demonstrating certain narrative themes, such as revenge or deception. Character schema can represent standard character goals such as pursuing romance, along with the characters themselves, their emotions, and physical objects. MINSTREL selects themes to represent in a story, and selects corresponding author schema. The author schema determine introduction scenes, denouement scenes, and thematic subgoals for the characters to solve in between.

MINSTREL employs a system called Transform Recall Adapt Methods (TRAMs) to find scenes in memory that fill these requirements and represent each theme of the story. The first TRAM searches the episodic memory for a sufficiently novel and appropriate scene, i.e. a scene fitting the thematic constraints that has not been used more than twice in previous stories. If no appropriate novel scenes can be found, TRAMs generate a new scene. Each TRAM contains a way of tweaking or generalizing an aspect of the schematic requirements, then searching for a scene fulfilling the new requirements.

That scene is used as an analog, and substituting the particulars of the current story schema into the pattern of the analogous scene results in a novel scene. After MINSTREL generates a story, it is added to the episodic memory database. Appendix A contains an example of a story generated by MINSTREL.

MINSTREL is a complex system capable of generating longer and more detailed stories than many previous systems, which can be subjectively verified by scanning Appendix A. However, Pérez y Pérez and Sharples express doubts about the robustness and scalability of TRAMs [Pérez y Pérez and Sharples, 2003]. They have proposed an alternative author modeling program called MEXICA [Pérez y Pérez and Sharples, 2001]. Like MINSTREL, it uses an episodic memory of schema from previous stories. Each action contains a tensional representation, a value showing how the event changes the amount of tension in the story. It uses a model of each character's emotions and experiences, called a story world context, but unlike procedural systems, has no set goal planning. Instead, MEXICA chooses actions from memory that are consistent with the acting character's emotions, relationships, and desires. If it cannot find an appropriate action, it enters reflection. In reflection, it evaluates the consistency, novelty, and interestingness of the story. MEXICA can improve consistency by finding actions with unfulfilled preconditions, then inserting actions before them to satisfy those preconditions. If the story is too similar to a previous story, future searches of memory will no longer choose events similar to those in that story. If the story does not have enough tension, MEXICA constrains searches of memory to events that increase tension. Essentially, MEXICA contains an internal representation of Ritchie's three criteria for creativity, and uses each as heuristics for choosing events from memory. Appendix A contains an example of a story generated by MEXICA.

The case based reasoning (CBR) system proposed in [Peinado et al., 2004] and [Gervás et al., 2005] uses a simplified version of author modeling. It uses stories based on Propp's folklore morphology, and includes representations of stories studied by Propp in its initial episodic memory. The user enters a query describing elements, such as characters or events, that should be in the story.

Then, the system uses an expansive semantic ontology to identify how each element fits into the morphology and find patterns in memory to which it can logically apply those elements. In other words, it finds an analogous case in memory and substitutes in the query values. To improve novelty, the CBR can combine multiple cases from memory, using certain morphological elements from different stories – for example, the villain from one story but the rescue plot of another – as long as the resulting story does not conflict with the logic encoded into the semantic ontology.

4.4 Drama Management

Another popular intermediate model is drama management. A drama management system is very similar to the basic procedural system: it is populated by various character agents, which use a goal-based planning process to determine actions. However, a centralized agent oversees the process and interferes if necessary. The purpose of this management agent is guaranteeing that the procedural system will generate an interesting and story-like narrative, thus solving the biggest problem of the basic procedural model.

Perhaps the most basic managed system is The Virtual Storyteller [Theune et al., 2003]. The overseeing agent, called the Director, has a set of rules identifying the beginning (establish characters), middle (characters try to achieve goals and come into conflict), and end (protagonist succeeds happily) of the story. The Director can reject character actions that would violate these rules, for example, not allowing the antagonist to kill the protagonist at the story's onset. It can also indirectly alter the course of the story to better match these rules by changing the environment or giving characters motivation. The Virtual Storyteller represents an even merging of procedural and declarative systems: the procedural side ensures character consistency, while the declarative side ensures proper story structure.

DEFACTO is an earlier yet more complex managed system [Sgouros, 1999]. DEFACTO generates interactive narratives by having the user take the place of a main character. A step in DEFACTO is a single interaction between characters. In every step, DEFACTO generates all the

possible actions between characters, i.e. all the actions that a character could take towards fulfilling its goals. Then, the Plot Manager evaluates the dramatic interest of the story state resulting from each. The dramatic interestingness of a story state is quantified by evaluating how closely the plot matches a number of "dramatic situation" patterns, such as rising action (represented as number of characters involved in the central storyline), reversal of fortune, or irony. Interactions involving the user are always more dramatically interesting.

Aylett et al. compare the role of a drama management agent to that of a human game master in role-playing games [Aylett et al., 2008]. Both are responsible for mediating the actions of independent agents representing characters, and both must guide the agents towards an interesting story without introducing illogical events or being overtly restrictive of agents' chosen actions. The paper outlines two possible implementations of such a system. In the first, character actions cause rule-based narrative "triggers" to fire, resulting in the execution of actions that advance a predetermined storyline. In the second, the management agent determines the outcome of character actions based on whatever will have the greater impact on the emotions of characters, effectively using net emotional change as a heuristic for drama and interestingness.

4.5 Interactivity Management

Drama management systems are not quite as easy to extend to interactivity as pure procedural systems. A management agent usually decides to take action based on the story world state that would result from that action, evaluating the interestingness or story structure appropriateness of the state. In a system where artificial agents control every character, the manager will have access to their planning processes and will be able to accurately predict the story world state that will result from an action. When a human user controls a character, it introduces an aspect of unpredictability to the system, preventing the manager from reliably predicting the result of actions. To compensate, an interactive system with drama management must contain a model of the user's knowledge in order to predict the

user's actions.

The IDA interactive drama manager was designed for interactive games with set stories [Magerko and Laird, 2004], but is applicable to narrative generation systems as well. The drama manager seeks to minimize points where narrative progression stalls or becomes impossible, in ways that the user will not recognize as limiting, manipulative, or otherwise out of the ordinary. As in one of the aforementioned "game master" systems [Aylett et al., 2008], the drama manager activates rule-triggered "plot points" once their preconditions are met. However, the drama manager will also use a model of the user's current knowledge of the story environment to predict how likely the user is to meet each of the preconditions of the next plot point. If the manager determines that it is unlikely for the user to meet the precondition, it will cause an audio or visual environmental cue to guide the user towards meeting the precondition. For example, a gust of wind to cause the user to look in the direction where an important diary is located. If it is impossible for the user to meet the preconditions, the manager will alter an unknown part of the story environment (i.e. something not contained in the model of the user's knowledge) to allow the precondition to be met. For example, if the user destroys an important key, the manager can create a copy in a room where the user has not yet been. The IDA drama manager helps ensure that a story stays interesting by working to keep the user engaged in a declarative or predetermined plot.

Nelson et al. propose an improvement to interactive drama managers based on plot points, such as IDA [Nelson et al., 2006]. IDA uses a search-based planning technique to determine the player's actions, searching through possible plans and determining which are more likely for the player to make. But since the search spaces can be intractably large in complex story environments, a reinforcement learning approach might be more effective. In temporal-difference learning, the estimated value of each state is dependent on its successors. If the dramatic value is comprehensively evaluated at the completion of a story, that value propagates backwards through all the previous states. In the technique proposed by Nelson et al., repeated simulations of an interactive story using a self-adversarial/self-

cooperative exploration (SASCE) player model "train" the state values. The SASCE player model has access to the drama manager's heuristic, and there is a chance that it will act either cooperatively (maximizing value) or adversarially (minimizing value), to represent the fact that users generally act cooperatively while still encouraging some exploration. At the expense of the initial investment of training the system, SASCE reinforcement learning can save a drama management system the trouble of dynamically predicting the user's actions and evaluating the resulting utility.

U-director is an all-encompassing system fulfilling both the interactive and narrative generation imperatives of the drama manager [Mott and Lester, 2006]. In each decision-making cycle, in which U-director can moderate character agents like The Virtual Storyteller or interact with the user like IDA, U-directer predicts the story state resulting from each possible action. It uses a Dynamic Decision Network, an extension of Bayesian networks that incorporates chance nodes, utility nodes, decision nodes, and time slices to model decision-making based on beliefs about a changing environment. The network incorporates the world model, a user model, and data about narrative flow, predicting the state of each both after the narrative action and after the user's predicted response. U-director evaluates utility based on the consistency of physical locations and character personalities, progress through the plot, relevance to current plot points, the user's independence (how much the director had to provide "guidance"), engagement (how often the user interacts with the world) and excitement (how often the user's beliefs change); all of which are part of the DDN. In other words, U-director takes consistency, interestingness, and the balance between narrative direction and user freedom all into consideration and makes an optimal decision as to what happens next in the story.

True Story is a system of generating narratives for persistent multiuser story environments [Pita et al., 2007]. It takes an approach inspired by (and thus applicable to) "massively multiplayer" online games, by breaking the overall story into a series of discrete "quests." True Story uses a combination of story roles, like LOGTELL, and interpersonal relationships, like many procedural narrative generation systems, to determine a character's goal in each quest. But instead of using a predefined

role like LOGTELL, True Story determines a character's role based on the character's capabilities and a stored memory of past actions. For instance, if a player has stolen objects, True Story might determine that he/she should perform a task befitting a thief. True Story also uses memory to determine whether a quest is relevant, and thus interesting and story-like, by whether its goals involve another character with whom past interactions are stored in memory. For example, if someone steals something from a character, the latter might develop a goal to discover who and tell the authorities, or take revenge. Since True Story was developed for multiplayer systems, it relies on human factors to make each quest interesting instead of evaluating the drama or story structure of quests. However, if the human players were replaced with procedural character agents, True Story would be able to generate a perpetual web of character goals, interpersonal relationships, and resulting intrigue.

The GADIN system (standing for Generation of Adaptive Dilemma-based Interactive Narratives) extends a drama-managed procedural narrative generation system to interactivity in an unconventional way [Barber and Kudenko, 2007]. The world model contains genre-specific locations, objects, and characters with non-deterministically determined traits, relationships, dispositions (emotional model), and beliefs. Like TALE-SPIN, actions are defined by a series of preconditions and postconditions, the former containing limitations on the traits and disposition of the acting character. Unlike most planning-based procedural systems, characters in GADIN simply act randomly within the constraints of their disposition and personality traits. Narratively interesting dilemmas are defined as situations that would be of different utilities for different characters, such as making a sacrifice (low utility for self, but high net utility) or choosing a character to favor (higher utility for one character vs. another). The GADIN narrative planner tests whether it is possible to insert a dilemma into the current story, and does so whenever possible, in order to add dramatic tension to the narrative. A random world property not present in the initial model but reachable through planning is chosen as the “story goal,” a condition to end the story that GADIN tries to “coerce” users to fulfill via the requests and information of other characters. The story goal must have some change in character agent disposition

as a precondition, which guarantees that the narrative will contain character development. GADIN uses an IDA-like evaluation system to determine whether it is unlikely that the user will reach the current story goal, in which case another one will be chosen. Unlike IDA and similar systems, if the user refuses to follow the story, even after hints, GADIN will change the story to accommodate him or her. However, in GADIN's current state, the story goals can appear arbitrary and the transitions between them disingenuous. Appendix A contains a sample story generated by GADIN.

5. Tools and Standards for Future Work

5.1 Development Environments

One of the greatest challenges facing narrative generation researchers is the amount of subprograms and resources that narrative generation systems require to produce results. A procedural system must have a fully operational planner and character agents, and a declarative system must have a great number of rules for decomposing a story. Most narrative generation systems must explicitly define every possible character action, with corresponding causes and effects. Furthermore, if the narrative has a textual presentation, the system will need vast semantic knowledge and language generation capabilities. If the system dramatizes the narrative through graphics, even more resources are necessary: extensive three-dimensional modeling and animation.

The production of these resources from scratch would require a great time investment. A development environment can provide some of these required resources along with a framework for integrating them. This allows researchers to avoid reinventing the wheel and focus more on original modifications and extensions. As an additional benefit, systems developed in the same environment will have certain commonalities that allow for easier development of testing environments (see Section 5.3).

Mimesis is a relatively venerable environment for the development and execution of interactive

narratives [Young, 2001]. It contains an interface with the computer game Unreal Tournament, which provides three-dimensional graphics, animations, and server capabilities. The narrative generation and story management system, Mimesis Controller, or MC, interfaces with a game server, the Mimesis Unreal Tournament server, or MUTS. The MC contains a variety of modules for integration into the narrative generation process. One module represents the story environment as a physical map, and another represents it as a knowledge base. Another module contains a planner that outputs commands to MUTS. These commands can be character actions, environmental changes, or system messages used in the communication protocol. The server contains a built-in "mediator" that acts like the IDA drama manager, detecting when user input would cause the MC-generated plans to fail and raising an exception, which it can either pass to the MC or act on a set of predefined exception-handling rules. The mediator can handle input by either causing the action to fail or altering the planned narrative.

Mimesis can support a wide variety of narrative generation systems. A procedural system could work by having character agent modules communicate with the planning module, or a more declarative system could generate a narrative through separate processes and allow the planner to handle low-level character actions, like in Facade. The exception system introduces a degree of interactive robustness to declarative systems, and is easily extensible towards a interactive drama management system.

Larios et al. have proposed an application programming interface, or API, that uses an object-oriented approach to the development and execution of narrative generation systems [Larios et al.l]. The API is object-oriented. Like Mimesis, it allows for three-dimensional interactive presentation of narratives. The graphics themselves are divided from the main program, so that they can easily be replaced when they become obsolete. The API provides a character agent object, complete with a graphical representation and a text-to-speech system. The character agent can be given predefined high level actions as commands, which it can decompose into animations, but it allows for low level development and recombination of new animations. A central process can give high-level actions to characters, creating a declarative system with reactive character agents like facade. The authors are

working to implement intelligent character agents by adding a goal-based planning system to the character agent object, which would allow for the easy generation of pure procedural systems.

5.2 Storytelling Improvements

The aforementioned development environments focus on narratives with interactive graphical presentations, each providing a way to translate high-level commands into animations. While unlike interactive animated narratives, prose narrative generation systems do not require any graphics at all, they encounter a comparable presentation problem. After such a system generates a story, it is obviously output as text. However, a simple text narration is likely to be unsatisfying to a human reader. Stories by human authors usually contain creative descriptions of characters and scenes, and feature character dialogue written using a dialect and vocabulary appropriate to the speaking character. They may relate events in a different order from that in which they happen, or withhold certain events from the reader to be inferred through context. In short, it is a difficult problem for a computational system to generate plausible human prose, never mind high-quality writing.

Certain narrative theories propose that narratives consist of three parts: the fabula, the suzjet, and the presentation [Callaway and Lester, 2002]. The fabula is essentially a knowledge base containing all information about the story world and its events. The suzjet lists which of the information or events will be told in the story, and in what order. The presentation of a prose narrative is the actual text of the story. Most narrative generation systems feature a strictly linear suzjet, stringing together every event in the fabula. Cheong and Young propose a more sophisticated way of determining a suzjet, to make a story more suspenseful [Cheong and Young, 2006]. Their algorithm, Suspenser, takes in a fabula (represented as a the output of a planner), a point in the story, and a value representing the desired suspense at that point (high or low). It evaluates the suspense of a point in a story as the inverse of how many ways a planner – and thus, theoretically, the reader – can find for the protagonist to achieve his/her goals. In other words, suspense reflects the direness of the protagonist's

straits. First, Suspenser determines which events are necessary for understanding the story. Then, it determines which of the remaining events raise suspense by cutting off preconditions in the protagonist's plans and which lower suspense by benefiting the protagonist. Suspenser composes a high-suspense suzjet by placing optional suspense-raising events before the given point, and suspense-lowering ones afterwards. To minimize suspense, the algorithm simply does the opposite.

Much as Suspenser creates a suzjet out of a fabula, Storybook creates text out of a fabula and suzjet [Callaway and Lester, 2002]. StoryBook is essentially a natural language generator meant for use with narrative generation systems. StoryBook uses an ontological description of the story world as the fabula. The suzjet is a sequence of events, potentially including scene and narrator changes, dialogue markers, and description requests. First, StoryBook uses a Narrative Organizer to make the suzjet into a hierarchical structure, roughly representing paragraphs and sentences. Then, by analyzing the frequency and recency of word usage, StoryBook replaces certain nouns with pronouns. Similarly, if a word is used too often, StoryBook replaces it with a synonym. Then, a sentence planner generates grammatically and thematically correct preliminary sentences and a revision process combines and rearranges them to prevent unnatural terseness. Finally, a surface realizer produces readable text. Appendix A contains a text generated by StoryBook, given a fabula and suzjet about Little Red Riding Hood.

Natural language generation can also aid narrative generation systems with interactive graphical presentations, by improving the quality of dialogue. The Hap character agent contains such a system, the Glinda natural language generator [Loyall and Bates, 1997]. The graphical presentation poses some challenge for good dialogue. Dialogue must work in tandem with the character's actions and expressions. It must not interfere with the character's ability to react to the environment, especially in an interactive systems. Hap represents language as an active behavioral process, an action that characters can take towards fulfilling a goal via communication. This allows characters to adapt or become distracted while talking if goals change or a more urgent goal arises. The tone of the language

in dialogue depends on the character agent's emotional state and personality.

5.3 Evaluation criteria

Perhaps the biggest problem hindering narrative generation research, even more so than the requirements of graphics or prose quality, is the lack of a means of comparison for evaluating the merits of different systems. There have been very few attempts to set a benchmark or standard for narrative generation systems. Most systems evaluate generated stories internally, if at all. Mateas and Stern, creators of Facade, proposed one of the first benchmarks in interactive narrative generation, but Charles and Cavazza largely discredit it [Charles and Cavazza, 2004]. The benchmark required a ten-minute long story with at least one discrete interaction per minute, which Facade was able to achieve. However, the benchmark does not consider the quality of the narrative at all. Furthermore, it does not consider how many narratives the system can generate, nor does it make any attempt at evaluating the originality of those narratives.

It is possible to discuss systems, as this paper does, in terms of Ritchie's creativity criteria of novelty, typicality, and quality, but there are no standardized quantitative measures of these criteria. Certain narrative generation systems evaluate these criteria internally, in ways that might be extensible towards narrative generation in general. The heuristic functions of drama management systems such as Virtual Storyteller and DEFACTO evaluate the adherence of a narrative to typical story structures, and thus are effective measures of typicality. Though it may never be possible to objectively quantify the quality of a narrative, the quality of art and literature having an inherent degree of subjectivity, many systems evaluate the interestingness or drama of stories as a measure of quality. Among the systems mentioned in this paper, MEXICA, Suspenser, DEFACTO, and U-director all evaluate interestingness to serve as a measure of story quality. Novelty is the criterion with the least amount of current progress towards evaluation. Divago, a model of human creativity that operates on the conceptual level, equates a novel concept as one not equivalent to any others to which the program had access [Pereira and

Cardoso, 2006]. Following this idea, it is very easy for an author-modeling or case-based system to calculate novelty by simply contrasting the structure and content of the current story with its database of previous stories. However, since other models of narrative generation do not contain such databases, or any representation of existing stories, the technique is not generalizable to these other models.

An external evaluation program might be able to compare the relative novelty of generated narratives by using a sufficiently large database of existing narratives, or even just of certain "cliché" storylines. With a large enough logical ontology of actions and their effects, the program might also be able to evaluate typicality, through an algorithm similar to DEFACTO's heuristic, and interestingness, through an algorithm similar to Suspenser. However, since the ontology and story database would have to be extremely large to be effective, as various narrative generation systems work across different genres and feature different actions, such a system does not currently exist. An alternative might be a dedicated study performed by a panel of human critics to rate narratives subjectively on novelty, quality, and typicality, but no extensive studies of this type exist either.

Recently, Peinado et al. have developed a testbed environment for procedural drama management systems [Peinado et al., 2007]. The testbed uses the Remote-Controlled Environments Interface (RCEI), a communications protocol and language. RCEI allows for an external testing environment to be synchronized with the drama management system's internal environment model. The external test environment is essentially a procedural narrative generation system. The drama manager can interact with it by giving imperative commands to the agents or environment, and the testbed reports back facts about actions taking place and their results. The testbed provides "challenges," in which the system must produce a specific storyline, or authorial goal, in the test environment without compromising realistic causality or the consistency of the given characters. The systems are scored on how often they are able to complete each challenge, with points for minimizing both time taken and amount of interference (i.e. imperative messages sent). Since the purpose of drama managers is largely to impose a story-like structure upon an otherwise procedural system with as little

interference as possible, this testbed seems like a promising way of comparing their versatility and effectiveness. However, the testbed has no representation of interestingness, limiting the scope of the testbed to drama managers that operate using more declarative methods of evaluating story structure, such as IDA. Also, the story quality and novelty of a system using a drama managers will still depend largely upon the nature of the procedural character agents.

6. Conclusion

To summarize, a narrative generation system nondeterministically produces and outputs a story, generally through prose or visually. Procedural systems, which generate emergent stories through the planning and interactions of artificial intelligence character agents, can produce a wide variety of stories with guaranteed logical and motivational consistency, though they may not produce an interesting or well-structured story. Declarative systems, which derive a story from the top-down, ensure a baseline of quality through adherence to storytelling conventions and, potentially, predetermined scripts. However, that also prevents their stories from being novel on a macroscopic level, and they adapt disingenuously to user interaction. Hybrid systems find a middle ground between the two, through author modeling injecting novelty into declarative story schema, drama management guiding procedural agents towards an interesting story, or another original approach. Currently, narrative generation research is in need of good development environments and evaluation standards.

Perhaps the most important recent trend in narrative generation is the shift to interactivity. As mentioned in the introduction, interactive narrative generation has more immediate applications than prose narrative generation. Because of the extreme gap between the quality of stories by human authors and those currently generated computationally, prose narrative generation has few immediate benefits, and risks being seen as trivial among computer science problems. On the other hand, human authors cannot implement dynamic interactive stories without writing out countless branches or

constantly communicating with the user. Interactive narrative generation is a novel means of entertainment, and has a very natural application to the growing video game industry. For example, a recently released video game, *Left 4 Dead*, uses a drama management system consisting of a central management agent called the "Director" overseeing character agents representing enemies [Valve Software, 2008]. Though the implementation details of the "Director" system are not public, it is meant to ensure an interesting and well-paced story that is different every time through, so it likely uses techniques that systems such as IDA and DEFACTO pioneered. The commercial success of such applications of interactive storytelling very well might lead to an increased interest in the narrative development research from which it came. Hopefully, with such an uptick in interest, high-level and prose narrative generation will be able to avoid the label of "novelty" or "toy problem," instead being recognized as an important landmark towards artificial intelligence and creativity

Appendix A: Narrative Generation Examples

TALE-SPIN [Meehan, 1977]

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

TAILOR [Smith and Witten, 1991]

Once upon a time there was an arctic tern named Truman. Truman was homeless. Truman needed a nest. He flew to the shore. Truman looked for some twigs. Truman found no twigs. He flew to the tundra. He met a polar bear named Horace. Truman asked Horace where there were some twigs. Horace concealed the twigs. Horace told Truman there were some twigs on the iceberg. Truman flew to the iceberg. He looked for some twigs. He found no twigs. Horace walked to the shore. He swam to the iceberg. Horace looked for some meat. He found some meat. He ate Truman. Truman died.

MAKEBELIEVE [Liu and Singh, 2002]

John became very lazy at work. John lost his job. John decided to get drunk. He started to commit

crimes. John went to prison. He experienced bruises. John cried. He looked at himself differently.

GESTER [Pemberton, 1989]

Charles lacked a city. As a result of hearing of Narbonne Charles wanted Narbonne. Then Aymeri agreed to help Charles. Then Charles and Aymeri rode to Narbonne. Then, Charles attacked the walls of Narbonne, currently controlled by Baufumez, helped by Aymeri. Thibaut and Clarion threw burning pitch down on Charles and Aymeri. Charles and Aymeri retreated. Then, Charles attacked the walls of Narbonne, currently controlled by Baufumez, helped by Aymeri. Thibaut and Clarion threw stones down on Charles and Aymeri. Charles and Aymeri broke into Narbonne. As a result of seeing Blancheflor Charles wanted Blancheflor. Charles succeeded in getting Narbonne. Charles praised God. Charles forgot to reward Aymeri. Charles threw Thibaut into prison. Then Charles planned to obtain Blancheflor for Charles. Then Aymeri refused to help Charles because he was not rewarded. Then Bertrand agreed to help Charles. Charles abducted Blancheflor, currently controlled by Thibaut helped by Bertrand. Because Thibaut was in prison he did not oppose Charles and Bertrand. Clarion opposed Charles and Bertrand in getting Blancheflor. Charles succeeded in getting Blancheflor. Charles praised God. Charles rewarded Bertrand.

Joseph [Beaubouef and Lang, 1998]

Once upon a time there lived a tsar.

One day it happened that Tsar went to the Palestine. When this happened, Tsartisa felt lonely for the Tsar. In response, Tsaritsa waited for Tsar to return. Some time passed.

Then King captured Tsar. When this happened, Tsaritsa felt lonely for the Tsar. In response, Tsaritsa made it her goal that Tsar would be rescued. Tsaritsa developed a plan. Tsaritsa rescued Tsar. This is how the plan worked. Tsaritsa disguised herself as the harpist. Tsaritsa went to the Palestine. Tsaritsa entertained King. Tsaritsa asked the King for the Tsar. Tsaritsa selected Tsar. Tsar was rescued.

BRUTUS [Bringsjord and Ferrucci, 1999] (Example found in [Pérez y Pérez and Sharples, 2003])

Dave Striver loved the university — its ivy-covered clocktowers, its ancient and sturdy brick, and its sun-splashed verdant greens and eager youth. The university, contrary to popular opinion, is far from free of the stark unforgiving trials of the business world: academia has its own tests, and some are as merciless as any in the marketplace. A prime example is the dissertation defense: to earn the PhD, to become a doctor, one must pass an oral examination on one's dissertation. This was a test Professor Edward Hart enjoyed giving. Dave wanted to be a doctor. But he needed the signatures of three people on the first page of his dissertation, the priceless inscriptions which, together, would certify that he had passed his defense. One of the signatures had to come from Professor Hart, and Hart had often said — to others and to himself — that he was honored to help Dave secure his well-earned dream. Well before the defense, Striver gave Hart a penultimate copy of his thesis. Hart read it and told Dave that it was absolutely first-rate, and that he would gladly sign it at the defense. They even shook hands in Hart's book-lined office. Dave noticed that Hart's eyes were bright and trustful, and his bearing paternal. At the defense, Dave eloquently summarized Chapter 3 of his dissertation. There were two questions, one from Professor Rodman and one from Dr. Teer; Dave answered both, apparently to everyone's satisfaction. There were no further objections. Professor Rodman signed. He slid the tome

to Teer; she too signed, and then slid it in front of Hart. Hart didn't move. "Ed?" Rodman said. Hart still sat motionless. Dave looked at him. "Edward, are you going to sign?" Later, Hart sat alone in his office, in his big leather chair. He tried to think of ways he could help Dave achieve his goal.

MINSTREL [Turner 1994] (Example found in [Pérez y Pérez and Sharples, 2003])

Once upon a time there was a lady of the court named Jennifer. Jennifer loved a knight named Grunfeld. Grunfeld loved Jennifer.

Jennifer wanted revenge on a lady of the court named Darlene because she had the berries which she picked in the woods and Jennifer wanted to have the berries. Jennifer wanted to scare Darlene. Jennifer wanted a dragon to move towards Darlene so that Darlene believed it would eat her. Jennifer wanted to appear to be a dragon so that a dragon would move towards Darlene. Jennifer drank a magic potion. Jennifer transformed into a dragon. A dragon moved towards Darlene. A dragon was near Darlene.

Grunfeld wanted to impress the king. Grunfeld wanted to move towards the woods so that he could fight a dragon. Grunfeld moved towards the woods. Grunfeld was near the woods. Grunfeld fought a dragon. The dragon died. The dragon was Jennifer. Jennifer wanted to live. Jennifer tried to drink a magic potion but failed. Grunfeld was filled with grief.

Jennifer was buried in the woods. Grunfeld became a hermit.

MEXICA [Pérez y Pérez and Sharples, 2001]

Jaguar knight was an inhabitant of the great Tenochtitlan. Princess was an inhabitant of the great Tenochtitlan. From the first day they met, princess felt a special affection for jaguar knight. Although at the beginning princess did not want to admit it, princess fell in love with jaguar knight. Princess respected and admired artist because artist's heroic and intrepid behaviour during the last Flowery-war. For long time jaguar knight and princess had been flirting. Now, openly they accepted the mutual attraction they felt for each other.

Jaguar knight was an ambitious person and wanted to be rich and powerful. So, jaguar knight kidnapped artist and went to Chapultepec forest. Jaguar knight's plan was to ask for an important amount of cacauatl (cacao beans) and quetzalli (quetzal) feathers to liberate artist. Princess had ambivalent thoughts towards jaguar knight. On one hand princess had strong feelings towards jaguar knight but on the other hand princess abominated what jaguar knight did.

Suddenly, the day turned into night and after seconds the sun shone again. Princess was scared. The shaman explained to princess that Tonatiuh (the divinity representing the sun) was demanding princess to rescue artist and punish the criminal. Otherwise princess's family would die.

Early in the morning princess went to Chapultepec forest. Princess thoroughly observed jaguar knight. Then, princess took a dagger, jumped towards Jaguar knight and attacked jaguar knight. Jaguar knight was shocked by princess's actions and for some seconds jaguar knight did not know what to do. Suddenly, princess and jaguar knight were involved in a violent fight. In a fast movement, jaguar knight wounded princess. An intense haemorrhage arose which weakened princess. Jaguar knight felt panic and ran away.

Thus, while Tlahuizcalpantecuhtli (the god who affected people's fate with his lance) observed,

princess cut the rope which bound artist. Finally, artist was free again! Princess was emotionally affected and was not sure if what princess did was right. Princess was really confused. The injuries that princess received were very serious. So, while praying to Mictlantecuhtli (the lord of the land of the dead) princess died.

GADIN [Barber and Kudenko, 2007] (Bolded lines are user's interactions)

- Billy is playing with Timothy.
- Timothy has a new toy, which he says is a time machine.
- He says that he will show Billy, says “Try to get ptero to like you!”, grabs Billy's hand and presses a button...
- Billy is in a forest! The forest is densely packed with trees.
- Billy can see diplo but can't see Timothy anywhere
- **Billy asks diplo to play**
- diplo decides to play with Billy
- Billy finds whaley's nest
- diplo is encouraging Billy to steal an egg from whaley's nest
- **Billy likes diplo and decides to agree to steal an egg**
- Billy now owns this egg
- ptero goes to the forest extremely grumpily and clumsily
- **Billy says ptero looks nice**
- ptero is pleased and becomes Billy's friend
- 'Wake up Billy!' It's Timothy. Billy is at home, and Timothy has just arrived.

StoryBook [Callaway and Lester, 2002] (From fabula and suzjet about Little Red Riding Hood)

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

One day her mother said, "My child, go to grandmother’s house. We have not heard from her for some time. Take these cakes, but do not stay too long. And, beware the dangers in the forest."

Little Red Riding Hood was delighted because she was very fond of her grandmother. Her mother gave her a well-filled basket and kissed her goodbye.

The road to grandmother’s house led through the dark forest, but Little Red Riding Hood was not afraid and she went on as happy as a lark. The birds sang her their sweetest songs while the squirrels ran up and down the tall trees. Now and then, a rabbit would cross her path.

Little Red Riding Hood had not gone far when she met a wolf. "Hello," greeted the wolf, who was a cunning-looking creature. "Where are you going?"

"I am going to my grandmother's house," Little Red Riding Hood replied.

"Ah, well then, take care in the forest, for there are many dangers." And then the wolf left.

Little Red Riding Hood was not in a hurry. Indeed, she gathered wild flowers and chased the pretty butterflies.

Meanwhile the wolf ran ahead very quickly and soon arrived at grandmother's house. He knocked on the door gently. The old lady asked, "Who is there?"

The wolf replied, "It is Little Red Riding Hood, your granddaughter."

And so the old lady opened the cottage door. The wolf rushed in immediately and devoured the lady in one bite. Then he shut the door and climbed into the old lady's bed.

Much later Little Red Riding Hood arrived at grandmother's house. She knocked on the door and shouted, "Grandmother, it is Little Red Riding Hood."

"Pull the string. The door will open."

And so Little Red Riding Hood opened the door and walked in. "Grandmother, what big eyes you have."

"All the better to see with, dear."

"Grandmother, what big ears you have."

"All the better to hear with, dear."

"And, grandmother, what big teeth you have!"

"All the better to eat up with!" yelled the wolf.

And then the wolf jumped up and devoured Little Red Riding Hood in one bite.