# The evolution of the fAIble system to automatically compose and narrate stories for children

A.J. Gonzalez, T. Anchor, A. Hevia, A. Posadas, J. Wade, R.A. Ansag, K. Benko, B. Bottoni, V. Kazakova, M.J. Alvarez, J.M Wong, J. Martin, R. Knauf, K.P. Jantke & A.S. Wu

**To cite this article:** A.J. Gonzalez, T. Anchor, A. Hevia, A. Posadas, J. Wade, R.A. Ansag, K. Benko, B. Bottoni, V. Kazakova, M.J. Alvarez, J.M Wong, J. Martin, R. Knauf, K.P. Jantke & A.S. Wu (2022): The evolution of the fAIble system to automatically compose and narrate stories for children, Journal of Experimental & Theoretical Artificial Intelligence, DOI: 10.1080/0952813X.2022.2104382

**To link to this article:** https://doi.org/10.1080/0952813X.2022.2104382

Published online: 25 Oct 2022.

Submit your article to this journal

Article views: 15

View related articles

View Crossmark data

JOURNAL OF EXPERIMENTAL & THEORETICAL ARTIFICIAL INTELLIGENCE
https://doi.org/10.1080/0952813X.2022.2104382

ARTICLE

# The evolution of the fAIble system to automatically compose and narrate stories for children

A.J. Gonzalez[a], T. Anchor[a], A. Hevia[a], A. Posadas[a], J. Wade[a], R.A. Ansag[a], K. Benko[a], B. Bottoni[a], V. Kazakova[b], M.J. Alvarez[a], J.M Wong[a], J. Martin[a], R. Knauf[d], K.P. Jantke[c] and A.S. Wu[a]

[a]Intelligent Systems Laboratory, Computer Science Department, University of Central Florida, Orlando, FL, USA; [b]Computer Science Dept, Knox College, Galesburg, IL, USA; [c]Adicom Group, Weimar, Germany; [d]Computer Science Faculty, Technical University Ilmenau, Ilmenau, Germany

**ABSTRACT**
This article describes our long-term research into automated story generation and our resulting story generation architecture called *fAIble* that incorporates several innovations. fAIble determines each event that occurs in the tale using a combination of scripted sequences and stochastically chosen events. The probability of an event occurring is based on the skills and personalities of the characters who have agency. Event selection is also influenced by the context of the situation faced by the characters. Each event is associated with a description in grammatically-correct natural language that can be narrated orally via text-to-speech. We describe the evolution of fAIble, its architecture and the results of our independent evaluation of each of the four progressively developed fAIble prototypes (fAIble 0, I, II and III), as tested with human test subjects. On a continuous scale where 0 means unacceptable, 1 means acceptable and 2 means optimal, the composite human test subject rating average from the independent tests of the prototypes was 0.933. The paper also describes a summative assessment where test subjects were asked to review stories from all four prototypes and rank them comparatively. These comparative results indicate an improvement from the original (fAIble 0) to the last one (fAIble III).

**ARTICLE HISTORY**
Received 13 October 2020
Accepted 17 July 2022

**KEYWORDS**
Narrative generation; automated storytelling; digital storytelling; automated story generation

## Introduction and background

Since before the dawn of written history, people have told stories to pass down values, teach lessons, and inspire the next generation. Tales, both true and fictitious, have been woven into art, philosophy and belief systems. Entertainment and education, therefore, have been two disciplines where story telling plays an important role, not only traditionally but also in today's world. It takes a special talent to compose stories that captivate a listener (or a viewer), not only to conceive an interesting plot with complex characters, but also to express the story in an equally captivating manner. As our appetite for more and better entertainment grows, the demand for more stories that are interesting has also increased.

**CONTACT** A.J. Gonzalez avelino.gonzalez@knights.ucf.edu Intelligent Systems Laboratory, Computer Science Department, University of Central Florida, 4328 Scorpius Street Orlando, FL 32816, USA

The rapid advances in computing, and specifically in Artificial Intelligence (AI) over the last 50 years has provided an intriguing way to fill this gap in story creation (Young, 2000). If AI could be used to its maximum advantage, interesting stories of arbitrary length could be composed and told in near-real time, and the stories could enjoy diversity, such that a subsequent story is significantly different from those previously generated by the system.

The computer science research literature contains many reports describing work to automate the story creation process. These have achieved varying levels of success, as many of the works report some success in generating stories that make some sense and are reasonably well structured, albeit with significant human intervention or preparation (*authorial burden*). However, a major breakthrough that is able to produce complete, long, complex, interesting, well-written stories that are equivalent to those of the best human writers has yet to emerge. While we certainly do not claim such a breakthrough here, it is in this pursuit that we base our work described here.

Before continuing to introduce and discuss our contribution towards this objective – a system we call *fAIble* – a brief discussion of the state of the art in automated story generation follows. We should note that it is not the objective of this article to provide an exhaustive review of the relevant literature. Instead, we provide a selected review of research by others upon which we partially base our work. For those readers who might seek a more extensive and detailed review of the field, we refer them to Ansag and Gonzalez (2019), which contains a recent and extensive discussion of the state of the art.

## Review of the relevant literature in automated story generation

Common threads throughout many of the recent automated story generation systems reported in the literature range from limited (or no) reasoning behind character actions to an over-abundance of repetitive sentence structures. Furthermore, there has been a growing community effort towards making the composed stories diverse so that subsequent stories generated by a system are significantly different from its previous ones. Lastly, most current systems require significant authorial effort on the part of the human 'author' in order to allow interesting stories to be composed. Therefore, our main goals in the development of the fAIble system was to build a storytelling system that could generate diverse stories, in well-expressed natural language, that would be interesting to hear, and all with minimal (or at least manageable) authorial effort from humans.

Many of the reported approaches focus on intelligent planning approaches to developing a plot – the sequence of top-level story events that cause 'motion' in the narrative. Other reported research works have focused on character development, where building rich and complex characters is the primary focus of the system. Yet other reported works focused on description of the story world and its elements. Such a definition of a complex and interconnected story world is particularly important in children's fairy tales.

One of the earliest attempts to automate the storytelling process was Meehan's Tale-Spin system in 1977 (Meehan, 1977). Tale-Spin used a predefined set of rules and world qualities to generate one of Aesop's fables. While it was very focused on this one goal and was not scalable to other applications, it did set the stage for future research in automated narrative generation.

Systems that followed Tale-Spin tended to cluster around the three areas of focus that we discussed above: 1) plot generation, 2) character development, and 3) world creation. The dominant approach has been to focus on the generation of an interesting and believable plot. Planning systems, many of which were developed as part of AI research have been the most popular approach in most of the systems that focus on plot generation. Early examples of such are MINSTREL (Turner, 1991) and IPOCL (Riedl & Young, 2010). More recently, some architectures have employed variations of the classical models of AI planning. These include heuristic planning algorithms such as those found in CONAN (Breault et al., 2021), and the adoption of LSTM networks, such as found in Plan, Write, and Revise (Goldfarb-Tarrant et al., 2019). Haslum (2012) argues that model compilation can be used to reduce story models that ' … go beyond the classical … ' AI planning models so that these classical models can in fact be used successfully to plan out a story's events. In his paper he proposes ways to accomplish this through the process of compilation. Porteous and Lindsay (2019) make the case that incorporating a process by which the antagonist continually seeks to interfere with the goals of the protagonist can reduce the authorial burden. They view this issue as non-cooperative multi-agent planning.

The second most common area of focus has been character development. Appropriate character development is crucial to good stories. Early systems such as UNIVERSE (Lebowitz, 1985) and BRUTUS (Bringsjord & Ferrucci, 1999) focused more on the characters in their stories than they did on the plot. Porteous et al. (2013) argue that the relationships among the characters in the story are as important as plot development, and that a robust set of relations among the characters can unburden the plot generation process.

The third (and smallest) cluster of focus in the research literature has been around building a world in which the characters live, move and act, and where the plots of the stories would unfold. Along with this is building the history (*backstories*) of the characters, especially as they relate to one another. These backstories can help explain conflicts between characters in a natural way. It can also serve as the foundation for flashbacks in the stories.

The computer game industry has created several interesting products that support automated story generation, specifically in background (world) generation. Caves of Qud (Grinblat & Bucklew, 2017) is a science-fantasy video game that generates unique biographies for major historical rulers in its world using Finite State Machines. Caves of Qud heavily inspired the approach taken for background generation used in fAIble. Dwarf Fortress[1] is a video game often praised for the granularity of its simulation process. What makes Dwarf Fortress so unique is the depth to which a fictional history can be generated. Nearly every aspect of its fantasy world is simulated. Development of Dwarf Fortress has been ongoing for more than 10 years, allowing the complexity and depth needed to achieve such results. The NPCAgency tool can enhance the depth of a story world through well-developed background characters (Pickett et al., 2015). It generates a suite of Non-Player Characters (NPCs) based upon a game universe's facts, rules, and constraints. The end product is an NPC with its own emergent history that is coherent with the story world. CONAN (Creation Of Novel Adventure Narrative; Breault et al., 2021) is a Procedural Quest generation system for quests given by NPCs in video games. CONAN aims to automate another realm of small-scale storytelling, one that can provide interesting quests for the player.

Of course, all three areas of focus are important when composing any excellent story. However, depending on the genre of the stories to be composed, one of the three may be more important than the other two. For example, character development could be said to be the most important in a love story, while plot development can be most important in an action-adventure-suspense thriller. For children's fantasy tales, however, we believe that world definition may be the most important of the three.

## Brief history of our project

This paper tells the story of how the fAIble system came to be – how it evolved and grew into its final shape over five years and four iterations. fAIble is the product of research supported by the US National Science Foundation under their International Research Experience for Students program, and in collaboration with our German partners at the Technical University Ilmenau, the Fraunhofer Institute for Digital Media Technologies and the Adicom Group. Four cohorts of four mostly undergraduate computer science students participated in this project, each for approximately 18 months. This included a two-month residence in Germany to work under the mentorship of our German partners, who are co-authors of this article. Each cohort (except for the first one, of course) took the work of the prior cohort and improved upon it. We begin by briefly describing a narrative generation system that was the predecessor of our work and the initial inspiration for fAIble – the *Campfire Storytelling System*.

### *Campfire*

The Campfire Storytelling System, built by Hollister (2016) as his doctoral dissertation, is a complete and automated narrative generation and storytelling system. The fAIble system was conceived to overcome some of the limitations of Campfire. We briefly describe Campfire here as the backstory to our main story, so to speak, and refer the reader to Hollister and Gonzalez (2019) and to Hollister (2016) for detailed descriptions, particularly the latter.

Campfire automatically creates quest-type fairy tales designed for young children as bedtime stories. It creates interesting stories of short (5–10 minutes) and medium (15 min) lengths (the user's choice). Campfire could be classified as a strongly plot-centric type of system. To help create the plot, Hollister conceived the *Cooperating Context Model* (CCM; Hollister & Gonzalez, 2018) as part of his research. CCM uses contexts to define a storyboard, which is later converted into the story plot. Campfire makes use of templates to dictate the major events in the quest story by implementing the seven phases of the quest (five phases for short stories), while CCM later determines the more granular events in each phase. Then, Campfire creates the natural language narrative to permit the story to be told to a child. Finally, it features a lifelike avatar that narrates the story out loud via a text-to-speech system.

However, the implementation of the system requires significant amounts of authorial effort by the user/author – its first drawback. The various types of characters (e.g., sorcerers, witches, peasants, knights, cowboys, rustlers, and space aliens among others), types of monsters (e.g., dragons, giant basilisks, etc.), vehicles (e.g., horses, spaceships, boats, etc.), weapons, treasures, and such must be pre-defined and included in its knowledge base. In all fairness, such world information would be necessary for any narrative generation system, and its acquisition could be difficult (although not impossible!) to automate. However, the templates themselves provide a challenge as they must be specifically designed for a type of story and are not trivial to build.

A second drawback to Campfire is that while it allows for several variables to be changed in the specification of the story (e.g., genre, characters, environment, assets, etc.), the overall progression of every story has the same seven (or five) stages of a quest. Thus, a listener who reads/hears more than say, three or four stories, even in different genres and with different characters and assets, will notice the repeating trend (although young children may not), even as the granular events may be different.

One notable and innovative feature of Campfire is that it permits interruptions by the listener as the story is being told to re-specify the created story. The system will modify the rest of the story according to the revised specification and will do it in near-real time. Interestingly, it determines the point in the story where the change in specification would fit most seamlessly, even when that point in time has already passed – sort of a primitive form of flashback. We should note that we did not seek to replicate this story modification feature in fAIble.

### *Introduction to fAIble*

The fAIble system evolved in four versions. We refer to these as AESOP (fAIble 0), fAIble I, fAIble II and the final version, fAIble III. Each of the versions introduced features that for the most part were retained and improved upon in the subsequent versions. However, in some cases, features that were not deemed to have worked well, as informed by the testing performed after each version, were eliminated. The final version is best described by discussing each individual version in detail. The next section introduces and discusses the first three versions, while the one after that discusses the final version.

## The early fAIble versions

In this section we briefly describe the first three versions of fAIble (AESOP, fAIble I and fAIble II). We begin with AESOP.

## AESOP

The first version of fAIble was actually not called fAIble originally, but rather AESOP (Wade et al., 2017), which stands for *Automatic Eclectic Story Origination Program*. AESOP was the result of a white paper approach taken by the first cohort to avoid using templates as did Campfire, as a way to truly diversify the stories generated. AESOP started out strictly as a storyboard system – that is, it sought only to create an outline of the sequence of events in the story (the *Fabula)*. This is a linear progression of actions in a story more akin to a list. It was subsequently modified to incorporate a minimal natural language description of the events so that it could be told as a story. Events in AESOP are described as simple subject-verb-object sentences.

Briefly, AESOP creates a *Story World* to log the state of the story and determine the possible actions to be taken next. It employs formal logic to represent all the elements. Its stochastic approach to event generation was designed to address story diversity. It also incorporates a degree of character development. It does not compose a backstory for the characters nor does it make much of an attempt at natural language generation.

AESOP builds its Story World by creating a 'bucket', so to speak, where all the Characters, Conditions, Actions and Objects are placed unsorted. This Story World reflects the state of the story – that which is currently 'true' at any one time in the progression of the narrative. More specifically, AESOP's story world is a collection of logic clauses (Conditions) that dictate the current state of the Characters and Objects. Characters can possess Objects, be situated in Locations, and have status (such as being alive or dead, being married to another character, etc.). Pre-authored content includes the initial characters, their attributes and relationships, an initial set of world conditions, and all possible actions (with their pre-conditions and post-conditions). Possible relationships are: friendship, propinquity (brotherly love), lust (romantic love), and loyalty. Possible attributes are strength, morality, and intelligence. There are three locations in the current prototype – RedHouse, Forest, and WitchHouse. RedHouse and WitchHouse are connected via the Forest.

A set of initial Story World conditions used in the AESOP prototype are:

```
isLocated(Harriette, WitchHouse)
isLocated(Witch, WitchHouse)
isLocated(Gary, RedHouse)
isAlive(Gary)
isAlive(Harriette)
isAlive(Witch)
isFamily(Harriette, Gary)
readable(Book)
has(Harriette, Cookies)
has(Witch, Book)
```

AESOP defines the *Story Space* as the universe of all conceivable Actions in the Story World (i.e., every Action (a verb) is permuted with every Character and Object that could act as its subject and predicate). This collection of Characters, Objects and Conditions, along with the Actions of the Story Space make up the Story World of AESOP. The Actions in the Story Space can then be filtered through contextually-derived pre-conditions into a *Possibility Space* that contains the actions that are possible at any one time given the current context. As the context changes, so do the actions in the Possibility State. Finally, the actual Action selected from the Possibility Space forms the next event in the story, and is assigned to the Fabula.

Figure 1 shows the concept of an expanded story world and the resulting output to the Fabula. The Current Space contains the Action that has been selected in the current iteration of the progressing narrative, and it is where the Action selection is outputted to the Fabula and to the story output as Subject-Verb-Objects triples.

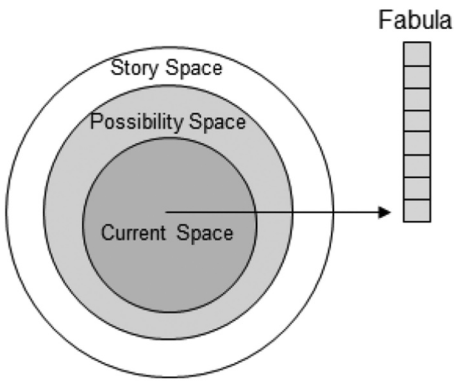


Figure 1. The expanded story World in AESOP.

With respect to event generation, AESOP selects the next event (action) stochastically from the Possibility Space, based on a probability influenced by *weights* assigned to attributes of the characters. These weights were selected heuristically by the development team based on their experiences, and were not scientifically selected or validated in any way. This obviates the need to plan the entire plot ahead of time and can serve to introduce surprising actions, something not possible in a template-based system such as Campfire. This also adds interestingness to the story and increases the likelihood of story diversity. Actions in AESOP have contextually-filtered pre-conditions and post-conditions. Pre-conditions dictate whether an action is possible, while post-conditions dictate how the world changes after the action is taken e.g., by removing or adding conditions to the Story World, as well as possibly changing a character's attribute and relation weights.

Any character can perform an action. There is no concept of a protagonist/antagonist in AESOP, and character agency is not the result of enabling a character in the story to take action. Rather, character agency is a number crunching process that matches every character's *personal parameters* (those strengths and weaknesses that define a character's persona) against the pre-conditions of a certain action and adding probabilistic weight to the 'likeliest' actions in the Possibility Space before the selection is made stochastically.

The personal parameters of the characters are reflected in their Attributes, Relationships and Goals. Attributes are parameters that describe a character's personal nature in isolation of other factors or characters, e.g., how strong or intelligent it is. Relationships on the other hand, represent personal qualities of the character that require an object or character to which the relationship applies – a character does not simply hate, it must hate another character. Each Attribute has a value and a weight indicative of that character's personal qualities, for example, a courageous character would have Attribute value = Courage; weight = 0.90. As an example of a Relationship parameter possessed by Harriette: the value = loves; object = Gary; weight = 0.95 indicates that Harriette loves Gary. Goals are essentially Story World state values that characters list as 'wanting' to be true. These goals are written a priori by the human author and assigned to the characters in the story before action generation begins. However, these Character Goals are not to be confused with Author goals, as the latter are more strategic and of higher level. Characters can either have or not have Goals. If character Goals are considered, then those actions that help realise a character's goals (or which make possible other events that may indirectly realise those goals) are made more probable. However, if character goals are not considered, then the next event is chosen strictly stochastically, without that consideration. These parameters (i.e., the values and corresponding weights in Attributes and in Relationships) help determine the next action chosen by the event selection system to become outputted and/or added to the Fabula. For example, if the state of the story presents a threat to a character, it has the option to fight, flee or hide. Depending on this character's attributes (courage, combat skills, fleetness of foot, stealthiness) one action would have a higher probability to occur than the other two. As an option, the author/user may specify via a Contextual Graph (see, Brézillon, 2004) – a sort of high-level storyboard – that sets up conditions that must be met before the story is allowed to end. For example, if revenge is Harriette's goal, it allows any sequence of events as long as Harriette ends up killing Gary. Possible actions in the AESOP prototype include:

@A kills @B
@A attacks @B
@A falls in love with @B
@A falls out of love with @B
@A seduces @B
@A asks @B to marry
@A agrees to marry @B
@A refuses to marry @B
@A befriends @B
@A becomes friend of @B
@A antagonises @B
@A becomes enemy of @B
@A meets @B
@A moves from @C to @B
@A eats @B
@A reads @B

However, there is more to event selection than just stochastically picking an event based strictly on the values and weights of personal parameters and/or goals as indicated above. There are ways to prioritise which parameters are the most important in selecting the next action (event). Six rule sets that determine the priority of these parameters (also called *modules* here) were created to affect the event selection algorithm as to how the event selection is to be made. While the use of more than one personal parameter at the same time (i.e., in the same run) is possible as well as desirable, the rule sets/modules are in fact mutually exclusive (i.e., only one can be used by the system for a complete story). They are listed below in order of increasing complexity. They are:

(1) **Random** – possible actions are equally weighted in terms of selection likeliness. No bias is given towards actions that are in line with character attributes or relationships. Impossible actions (e.g., characters taking action after dying) are not pruned out. This is essentially a control situation for the system to determine whether the story action selection algorithm that makes use of selective probabilities could in fact produce more meaningful stories than the virtual gibberish sure to be generated with this module.
(2) **Possible Actions** – Almost the same as the **Random** module above, except that impossible actions are pruned out.
(3) **Character Attributes** – An upgrade over module #2 above, where actions are now weighted towards character attributes. If a character has a low Morality attribute (i.e. character X has a Morality of 10/100), AESOP is more likely to select actions that have a lower Morality precondition where that character acts (i.e. X kills Y).
(4) **Character Relationships** – Same as module #2 above, with the addition that actions are now weighted towards character Relationships. If a character has a bad relationship with another (i.e. X's affection towards Y has a value of –75/100), AESOP is more likely to select actions with a lower character relationship precondition where that character acts (i.e. X kills Y).

(5) **Character Goals** – Same as #2, with the addition that characters now have Goals that are taken into account in event selection. Weight is added to actions that would make the goals come true, either directly or indirectly.
(6) **Context –** the same as #2, with the addition that there are 'author' goals that are specified in a linear Contextual Graph – a type of high-level storyboard that lists a series of three world states that must happen in sequence – referred to as chapters 1, 2 and 3; however, the specific Actions which are actually selected are generated using the AESOP event selection algorithm. Each 'storyboard' can be imagined as a series of steps where progression to the next context occurs when specific events have happened. These storyboards are different from the templates in Campfire because unlike those in Campfire, they do not dictate the actions to take place next – the AESOP stochastic event selector still does that. It simply allows an author to enter goals for the stories that have to be satisfied, although not linearly as in Campfire. These contextual graphs are created a priori and assigned to the story before any action generation begins. The AESOP prototype contains two such modules – the *Revenge* context and the *Change of Heart* context.

The use of these modules, we believed, would create stories that struck a balance between unpredictability and predictability. Actions that were in line with either character relationships, character attributes, or character's goals were weighted to be more likely depending on which module was enabled. However, the action selection algorithm still runs stochastically with a random number generator, and could select unlikely actions.

With regards to character development, relationships between characters are also explicitly defined and similarly quantified. These include a valence (negative or positive) to indicate things like love and hatred. Characters do have the goals discussed earlier. The values and weights of characters' attributes and relationships can and do change throughout the story as a result of the post-conditions of actions taken in the event generation step. The definition of each action includes the 'ideal' attributes and relationships that are in place for the action to occur. The closer a character's actual attribute/relationship values are to the ideal values, the more likely that action is to be selected as an event. If that action does occur, then the character's attributes and relationship values and weights are modified to be closer to those of the ideal values. If the action had a probability P of occurring ($0 \leq P \leq 1$), and for a given attribute or relationship weight, the character's weight X is modified to be closer to the ideal value V as follows:

$$X = X + [(1 - P) * (X - V) * \text{modifier}]$$

As the new attribute/relationship weight experiences greater change the less likely the action will be. A modifier (hard-coded by the programmer) makes the change either more or less drastic. The modifier value in the prototype is 0.2, which was determined experimentally via trial and error.

AESOP does very little towards natural language generation (NLG). It describes its output events very simply in a subject-verb-object format. This is because AESOP was initially designed to only build the Fabula, rather than a full listener-ready narrative. It was only much later that it was modified to generate some semblance of a listener-ready narrative. Unfortunately, the resulting story text is rather terse and unexciting, leaving out the details of how or why what happened, but this was its original mission.

Below is a sample story generated by AESOP. It is in the context of a Hansel and Gretel type of story. The names were changed to Gary and Harriette. As can be seen from the story below, AESOP's output comes only as a block of text to be read to (or by) the child.

Witch reads Book. Witch meets Harriette. Harriette antagonizes Witch. Harriette becomes an enemy of Witch. Witch seduces Harriette. Harriette becomes a friend of Witch. Harriette asks for Witch's hand in marriage. Harriette eats Cookies. Witch agrees to marry Harriette. Harriette seduces Witch. Harriette befriends Witch. Harriette attacks Witch. Harriette kills Witch. Gary moves to Forest from Red House. Gary moves to Witch House from Forest. Harriette meets Gary. Gary becomes a friend of Harriette. Harriette becomes a friend of Gary. Gary antagonizes Harriette. Gary becomes an enemy of Harriette. Gary attacks Harriette. Gary attacks Harriette. Harriette attacks Gary.

Note that while a storyboard is indeed created, the events above are described in very short sentences making them too terse to be interesting. Nonetheless, note how some of the events are contradictory, others are repetitive, and yet others are trivial in nature. Furthermore, there is no explicit world description and the causes and consequences of the events are not explained. Nevertheless, it did tell a story, albeit not a very good or interesting one.

For additional information on AESOP, please refer to Wade et al. (2017).

## fAIble I

The second version of the system was called fAIble I (Kazakova et al., 2018), as the name AESOP had led to confusion with Aesop's fables themselves as well as with another system in the literature with the same name. The name fAIble is not an acronym, but does include and highlight 'AI' within its name to indicate its underlying technology. fAIble I's main objective was to address the problems found in AESOP and create naturally-sounding stories that maximise variability by focusing on modularity and reuse, and eliminate path-finding plans by incorporating deus ex machina fixes to any dead-ends that may arise in the narrative. Several of the concepts found in AESOP remained as part of fAIble I, namely stochastic event selection. However, in lieu of the formal logic representation found in AESOP, fAIble I incorporated a graph database for keeping track of context manipulations and the choices that the characters consider throughout the narratives. All actions were augmented with probabilistic *interrupt* events that would present obstacles to the protagonist along its way through the current quest, incorporating side-quests, and allowing for a high number of permutations of events, resulting in less repetitive narratives. Additionally, a separate Natural Language Generation (NLG) layer was added to process the narrative events and improve the quality of the generated text.

### Graph database implementation

Similarly to AESOP, fAIble I implicitly creates a story world that stores all physical and abstract elements involved in the narrative. This story world also incorporates basic quest types as minimalist high-level templates, and common sense concepts as minimalist high-level decision graphs, where specifics such as weapon availability or individual morality are plugged in at decision time. The world and its state are now represented in a graphical database for greater efficiency, ease of traversal, and a more developer-friendly extensibility, given its plain-English format and native visual representation of all elements, their attributes, and interconnections, as opposed to the logic clauses in AESOP. This was a major innovation of fAIble I that persisted through the subsequent fAIble versions (II and III).

fAIble I uses the neo4j graph database. Our goal was to capture high level common-sense knowledge and reasoning, with a focus on fairytale structures (quests, roles, etc.). The relevant concepts and their relations can be intuitively seen as Label Property Graphs (LPGs), the model behind neo4j. LPGs are designed for storage and efficient query by focusing on objects and their relations, as well as on the properties of both (i.e. object properties and relation properties). This results in a significantly more compact graph representation of the domain, as much of the data are neatly tucked away under vertex and edge properties, allowing for simpler and faster queries. The Cypher query language directly follows the relations/edges as pointers throughout the graph, something necessary for real-time narrative generation. LPG is also of particular utility to the design of a system for automated narrative generation, as it provides a clear view of the general contexts in which the characters find themselves, as well as the particular properties of these contexts that may influence their feelings and decisions. Thus, the intuitive and compact representation also helps with the analysis of system capabilities and shortcomings, facilitating revisions and extensions. See, Figure 2 for a high level graphical depiction of how fAIble I creates and uses the story graphs.

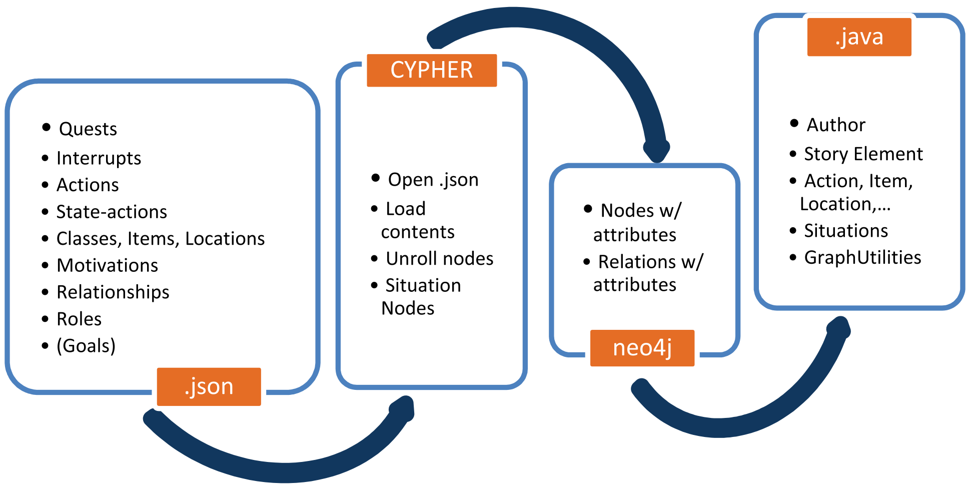


**Figure 2.** High-level view of the operation of fAIble I.

Other knowledge representation paradigms were considered, but a graph database was ultimately chosen as the most suitable. RDF and OWL languages represent knowledge as large graphs of atomic facts, making them well suited for unstructured data and automatic knowledge extraction for NLP tasks. While these languages facilitate data exchange and making rule-based inferences from large sets of data triples, they are not designed for direct human consumption, as would be necessary in our application. While automated inferencing has many benefits, querying data stored as RDF triples requires many joins and can, therefore, require queries that are complex to design and slow to execute, making them less than ideal for online narrative generation. Overall, breaking down the narrative concepts, relations, and their properties into atomic RDF triples would result in substantially larger graphs that are more complex and slow to query, as well as more difficult to analyse and improve.

While neo4j can import data from a variety of formats, we chose to work with JSON, as its minimal syntax provided the cleanest and most human-readable format for our manually authored files. These files include lists of all possible character classes, items, locations, motivations, goals, relationships, and feelings, as well as some connections among them (e.g., the items a princess is most likely to have in her possession at the beginning of a story).

The graph database of narrative elements includes a number of sub-graphs, each corresponding to aforementioned JSON files. For example, one sub-graph represents all possible character classes, while another represents all possible items. Additional sub-graphs were created to represent more abstract narrative elements, such as states, actions, and interrupts events. Each node in the graph has attributes (such as weight and colour) and relationships to other nodes in the same or other sub-graphs. For example, character classes can have a 'Likely accompanied by' or 'likely accompanies' relationship with another character class node within our classes sub-graph, as well as a 'likely possesses' relationship with item nodes in our items sub-graph. These relationships can also have any number of attributes, allowing the quantification of feelings, likelihoods, etc. In this way, the nodes and relationships in the graph database are highly expressive as well as flexible, easily adapting to current and future system needs.

## *Narrative generation*

fAIble I generates narratives from minimalist high-level quest templates (composed mainly of goals and roles) and allowing a protagonist to navigate from its starting state to the goal state by following a graph of common-sense high-level considerations, which incorporate contextual specifics to arrive at context-specific conclusions. Variability is further increased by allowing actions to fail, each failure leading to new state and possibly interjecting existing goals with side-quests, either optional or mandatory with respect to a protagonist's current goals. The same mechanism was also intended for narrative adjustments based on reader/listener requests, but was not implemented in the prototype.

Like AESOP, fAIble I does not employ any kind of path planning, but unlike AESOP, it is primarily goal or sub-goal (in the case of sub-quests) driven. Path-planning can lead to dead-ends, but in human-created narratives any dead-end can be avoided through creativity. This was a major goal of fAIble I: to allow for any path to reach a resolution by fixing broken world conditions as they arose. For example, if a character were to get stuck at an impassable bridge, a newly generated object, character, or piece of information shows up to create a new path to the goal and thereby resolve the potential dead-end. The prototype incorporated paths from failed actions to put the protagonist back on track based on common-sense actions and updates to the character's current conditions. Additional deus ex machina elements were planned (such as allowing addition, removal, or change to any story element by reader/listener request). but these were not incorporated in the prototype, as no further dead-end options were possible at that stage of the prototype.

Every action taken by the protagonist is allowed to fail, forcing the character to try again or to find an alternative way to keep moving towards its goal. Such interrupts affect the most expected goal- and personality-driven behaviours, and serve to diversify and complexify the basic plot. Each time an interrupt changes the character's context, a new sub-goal may be added. Such sub-goals can be mandatory and require completion before continuing towards the current goal (e.g., if it wants to get to the other side of this lake it will need to find a boat), while others may be optional (e.g., it sees a 'Wanted dead or alive!' poster and can choose to add the side-quest of finding this person to its other ongoing goals, or continue without doing that). The nature of how this quest is related to the existing quests will impose some partial ordering on future choices. At each narrative chapter, the character will select one of the goals it is working towards (among those which are not currently blocked by other more immediate goals), and then take some number of steps towards completing that goal.

Like AESOP, it can be said that fAIble I is a plot-centric system. The event selection process also works similarly to how AESOP selects the next event (action), except that fAIble I includes the *thinking-acting* process, stored in the system as the Thinking-Acting sub-graph. One action is selected from a set of possible situationally-relevant actions. Some actions are context-specific while others are universal (context-free) and, therefore, always available. Actions that modify the current state most favourably with respect to the goal are given higher probability; this allows any feasible action to be selected, but tends towards the most sensible goal-oriented behaviour. This tendency can of course be modified based on character intelligence. The character's morality and feelings will result in some actions being preferred over others, with opponent's morality also being accounted for during decision making. A character's aggressiveness or persuasiveness can also be incorporated as modifiers, although these adjustments were not incorporated into the fAIble I prototype.

In the fAIble I prototype, only the protagonist is given a starting goal and agency to pursue it, though our research goal is for all main characters to ultimately have agency (fAIble II accomplishes this). In order to maximise the reusability of all elements within the system, the fAIble I prototype was supplied with general definitions of the most common but minimally defined calls to action: quest for an item (either to save a loved one or to recover a stolen possession) and quest for a character (either to exact revenge, find a loved one, or to make some request), as these tend to cover the vast majority of quest goals. These general definitions are then supplemented with specifics, either at random from the set of all available elements within the system or by user request. For example, if an item was stolen, then we must specify what item, indicate that the protagonist was initially in possession of this prized item, and then specify who stole it to provide the protagonist with an item recovery quest.

A character's *thinking-acting* cycle begins with an abstracted goal state, for example, 'Have <item>' and an opposing start state of 'Not have <item>'. From there, logical thinking dictates that a character consider whether another character has the item and, if so, to then add a new sub-goal of 'Be at location of <other_character>'; otherwise, the next desired state would be 'be at location of <item>'. Then the character must assess whether it is already at that location or whether it needs to travel to it. Once co-located with the other character, it must consider options of dialogue, violence, subterfuge, etc. The character will employ the options that are most likely to lead to the desired outcome based on its own characteristics of persuasiveness, fighting skill, morality (both its own and of the other character), etc. To provide our characters with the largest number of possible actions, fAIble I lets them assess their options based on basic characteristics, such as the weight of the items in their inventory. For example, while attacking with something that is considered a *weapon* is most sensible, attacking with something heavy is also a reasonable option if no *weapon* is available. In this way, a princess might not have a sword, but may still hit someone with her heavy crown. The character's strength and dexterity determines the likely outcome of a potential attack, and is weighed against the possibility of convincing the other character to hand over the item based on the character's persuasiveness value instead. The outcome of the chosen action will update the current state with respect to the goal. Maybe the protagonist will now have the item, indicating that its goal or sub-goal has been achieved, or the attack fails and the other character escapes, thereby forcing a new sub-goal of finding it again; or maybe the other character will fight back, potentially affecting its dexterity and thus changing the next set of viable actions. The next set of decisions will depend on the protagonist's state once again.

With regards to character development, fAIble I retains the identification and quantification of attributes used in AESOP. These are called *physical and psychological traits* in fAIble I. These values also influence the probability of a specific action to be selected from among those actions that are situationally relevant. However, unlike in AESOP, the relationships are established as a hierarchy of effect, where feelings between characters determine the nature of their current emotional relationship, allowing these relationships to evolve over time. Relationships between specific characters can change as a result of interaction-based shifts in characters' feelings towards one another, thus allowing enemies to become friends and vice versa. For example, if the protagonist hurts someone or steals from another character, the affected character's feelings towards the protagonist will grow more negative. The prototype implementation of this is somewhat limited, but it contains the groundwork for incorporating sentiment alongside character attributes into the contextually-delimited action selection.

As characters make decisions regarding how to reach some desired state from their current state, they traverse the *thinking-acting* graph. As all choices come from specific pre-action considerations or 'thoughts', these can be shared with the reader in order to explain the character's actions. The fAIble I prototype translates these pre-action considerations into text, as an initial attempt to allow the reasoning to be more transparent and appear more sensible to the reader/listener. While this rudimentary solution resulted in less than elegant narratives, with further pruning of some decisions, highlighting of others, and additional Natural Language Processing, the system could deliver more fluid and natural insights into character thoughts. This transparency could be used to teach children thinking logic and how to balance the means versus the ends.

## *Natural language generation*

fAIble I introduces a dedicated Natural Language Generation (NLG) module to improve on AESOP's rigid and terse narrative content by focusing on language diversification. Along with the graph database, the NLG module is the most significant upgrade over AESOP. The NLG module is built on top of SimpleNLG (Gatt & Reiter, 2009), an NLG engine used for composing sentences. The NLG module allows fAIble I to implement three important processes by off-loading the responsibility of narrative language generation to the NLG module. First, it incorporates character reference diversification, by changing how a character is mentioned through the use of names, titles, and pronouns.

Second, it implements event aggregation, minimising repetitive events that do little to move the narrative forward. Third, language is embellished with contextual knowledge of the characters' attributes, morality, and feelings. Taken together, these processes allow for more language variation and overall richer, more descriptive narratives.

***NLG: Reference variation.*** The fAIble I NLG module applies the first process by incorporating varied use of common nouns, pronouns, and proper nouns. More generally, it is able to define characters by various characteristics including names, roles, titles, and gendered pronouns. For instance, it can refer to a character named Daria by her name, by her title of 'mercenary', or by her gendered pronouns 'she/her'. This reduces narrative redundancy by preventing the generation of sentences such as, 'Daria set out to avenge Daria's mother'. To this end, fAIble incorporates pronominalisation, i.e. the use of pronouns, as opposed to the exclusive use of nouns as subjects or objects of sentences. For example, 'Mary ran to Ali's house' is not pronominalised while 'she ran to his house' is. Of course, Mary and Ali would have to have been mentioned earlier and in close proximity to this statement. Moreover, referring to characters by their titles increases variability in sentences. Which reference is used (name, pronoun, title) is determined using a weighted probability where each reference is initially equally likely, and that likelihood drops depending on how recently and how much that reference has been used. Characters are always referenced by name when first introduced. Furthermore, upon first appearance, titles (e.g.,'princess') first come preceded by indefinitely articles ("*a* princess), followed by definite articles in all subsequent mentions (*the* princess). Scene *transitions* (defined by a change in location) reset all likelihoods for reference variation (but the definite articles remain).

***NLG: event aggregation.*** Event aggregation is the second process done by the NLG module. It combines similar events to make the narrative sound more natural. An example of a battle scene created by fAIble I illustrates why aggregation is desirable:

> The mercenary struck the guard with her dagger
> The guard grabbed at the mercenary, but missed
> The mercenary struck the guard with her dagger
> The mercenary struck the guard with her dagger
> The mercenary struck the guard with her dagger
> The guard fell over and died

'The mercenary struck the guard with her dagger' is repeated multiple times, adding little to the story, and increases narrative redundancy (and potentially a reader's boredom), substantially decreasing the quality of the narrative. Event aggregation fixes this by aggregating events without loss of meaning:

> The mercenary struck the guard with dagger
> The guard grabbed at the mercenary, but missed
> The mercenary struck the guard with her dagger 3 times
> The guard fell over and died

Given the probabilistic context-based nature of fAIble I's action selection, coupled with the small space of logical choices implemented in the prototype, it is possible that similar (or even identical) actions could take place in short succession. To address this, the NLG module processes all Actions in a Scene prior to generating the narrative content pertaining to that Scene. It then determines how similar successive Actions are based on the *actors* and *recipients* of the Action, as well as the verb itself. If two successive Actions have the same *actors, recipients*, and *actions*, the NLG module aggregates them in the way shown in the previous example. If two successive Actions have the same *actors* and *recipients* but different *actions*, the *actions* are combined to create a compound *action*. Thus, 'Mary attacked the wolf. Mary jumped on the wolf.' becomes 'Mary attacked and jumped on the wolf.' Similarly, if the two Actions have the same *action* and same *actors* but different *recipients, recipients* are combined to create a compound element. Thus, 'Mary attacked the wolf. Mary attacked the hog.' becomes, 'Mary attacked the wolf and the hog'. While these examples were limited to two successive Actions, the same process can be extended to an arbitrary number of successive Actions.

***NLG: character-specific descriptor enhancement.*** The characters in fAIble have attributes that determine how well (or poorly) they perform certain actions. These attributes range from 'dexterity' to 'charm'. To diversify narrative content, the NLG module leverages these attributes to enhance descriptions of character actions with adverbs. For example, depending on a rogue's proficiency in 'dexterity', he may 'clumsily' or 'deftly' pickpocket the guard (or perhaps even fail altogether), while a brave and dexterous knight is more likely to act 'valiantly' and have descriptions of his 'fearsome' swordsmanship.

fAIble I's Story Generator incorporates attributes and labels into elements in its Story Space to facilitate adverb enhancements. The Story Generator tags each Action with a label based on the type of action it is (e.g., 'skulking' may be tagged with the label of 'sneaking'). Additionally, fAIble I has a word bank consisting of adverbs that are grouped into categories determined by the level of 'skill' they may require. For example, 'swinging' a sword at someone may not require the same level of strength as 'hacking' at someone with a sword. This word bank also specifies to which character attributes the adverbs correspond. When the NLG module processes an Action, it calculates the level of 'skill' the character employs when carrying out the action. This calculation is based on the character's proficiency in the attribute that this Action uses, the aforementioned label of the Action, the history of adverbs used so far (along with how frequently they were used), and an element of chance (there is always at least a 10% chance an action would not be described using an adverb in order to not overuse adverbs). Unfortunately, the time restrictions imposed in the development of the fAIble I prototype required that the word bank and the adverb associations would have to be built manually. Future versions of fAIble will make use of semantic mapping tools that use WordNet (Miller et al., 1990) to increase the dynamism and fluidity of adverb enhancements.

Other features of fAIble I include incorporating character pre-dispositions and other characteristics into the narrative as a way to indicate surprises when a character's actions go against these pre-dispositions. This would enhance reader engagement. For instance, when describing a generous act by an old miser, the action could be phrased as, 'Surprisingly, the old man gave the beggar all of his copper'.

The fAIble I NLG module has its limitations. For one, it does not incorporate segue generation, so transitions generated by the NLG module are abrupt, forcing the reader to quickly switch contexts. This can decrease reader satisfaction and cause confusion. Additionally, the Thinking-Acting graph was disconnected from the NLG module in the fAIble I prototype. Thus, character 'thoughts' were static strings generated by the Thinking-Acting graph, leading to unnatural 'thought' patterns appearing throughout the narration, such as: 'Nathaniel thought, "Is Nathaniel evil?". "No", thought Nathaniel'. Allowing the NLG module to re-process this content would have allowed for a more natural flow throughout the narration and helped alleviate the issue of abrupt segues.

Finally, we should note that fAIble I, as does AESOP, only outputs a block of text that must be read to (or by) the listener (likely to be a child). A story generated by fAIble I follows:

> Our story begins in a village. A mercenary named Nathaniel lived in the village. The village was welcoming. Nathaniel had a companion named Christopher. Daria, the creature, killed Christopher. Nathaniel set out to avenge Christopher. Nathaniel thought: "Is Nathaniel at location of Daria: forest?", "No" he thought. Nathaniel went to a forest. The forest was sunny. Nathaniel thought: "Is Daria dead?", "No" he thought. Nathaniel thought: "Is Nathaniel evil?", "No" he thought. Nathaniel thought: "Is Daria evil?", "Yes" he thought. Nathaniel thought: "Nathaniel has weapon?", "Yes" he thought. Nathaniel attacked Daria violently with a sword. Nathaniel tried to attack Daria with a shield but failed. The creature escaped to a road. Nathaniel thought: "Is Nathaniel at location

of Daria: road?", "No" he thought. Nathaniel went to the road. Nathaniel tried to attack Daria with the shield but failed. Nathaniel attacked Daria violently with the sword. Nathaniel killed the creature. Nathaniel thought: "Is Daria dead?", "Yes" he thought. The End.

While not perfect, note the significant improvement seen in the above narrative from the AESOP sample story. Unlike AESOP, which only generated very short events often with little or no transition, fAIble I actually generates story events (albeit mostly only at a high level) that are linked logically. The language used in fAIble I is also much more natural than the staccato-like expressions found in AESOP. However, the excessive and awkward thought revelation can be rather distracting and annoying. For a full description of fAIble I, please refer to (Kazakova et al., 2018).

## *fAIble II*

The third iteration of fAIble – fAIble II (Alvarez et al., 2019) – primarily sought to enhance the character development capabilities of the prior systems by permitting them to have emotions, both short and long-term. In the process, the system's modularity was enhanced to allow for independent editing and debugging of each part of the process, as well as for better scalability. The knowledge graph in fAIble I stores the relationship of possible story arches; in fAIble II, the narrative was also moved into the graph. To clarify, the structure of the graph was used to drive the story in fAIble I; however, in fAIble II the story itself and everything about it is in the graph. Thus, generating a story became a set of queries and mutations to the story graph.

### *Narrative generation*

Event selection is still largely stochastic in fAIble II; however, the trend begun in AESOP and to a greater degree in fAIble I to use character goals to influence the next action to be selected was extended, as fAIble II uses characters' goals as weights when deciding on the story's next event. In the process, fAIble II contains a graph representing every character's goals, and queries this goal graph when deciding on the next event.

Additionally, agency was granted to more than just the protagonist, and a turn-taking system was implemented to alternate which character had agency to act. One major enhancement was the addition of an *Event Translation* module that takes place right after event selection to filter the events and give them more structure before they were passed to the NLG module. Finally, another of the salient innovations of fAIble II is the *Mind Graph* – a sub-graph that represents the character's situational awareness at any point in time. The Mind Graph – not to be confused with the Thinking-acting graph of fAIble I – can and does change as the situation around the character changes, and this can lead to character development.

A second significant enhancement of fAIble II is that it explicitly introduces world creation as an explicit step in the narrative generation process, rather than simply being implicit parts of other steps as manually-populated repositories of general knowledge, as done in AESOP and fAIble I. The story-making process now has an explicit step where characters, locations and objects that will be part of the story are defined and described semi-automatically. Furthermore, the world now has an explicit map, as the specific locations are reflected in the graph database. Locations are classified as *abstract* (e.g., France) or *specific* (e.g., the Eiffel Tower in Paris). Links between the nodes reflect the paths that characters can use to physically transport themselves between specific locations. These links also define the means of transportation available. Graphs containing the characters and assets to be used in the story are also part of the created world.

Characters can have roles, titles, gender (for sentence generation) and basic health statistics. Upon character creation in the World Generation stage, the *Mind Graph* is created and attached to the character's node. The Mind Graph is a sub-graph that serves as the driving force behind character development – its perception, emotion, personality – and eventual story progression. It is composed of nodes that represent the character's understanding of entities in the story world. Thus, the Mind

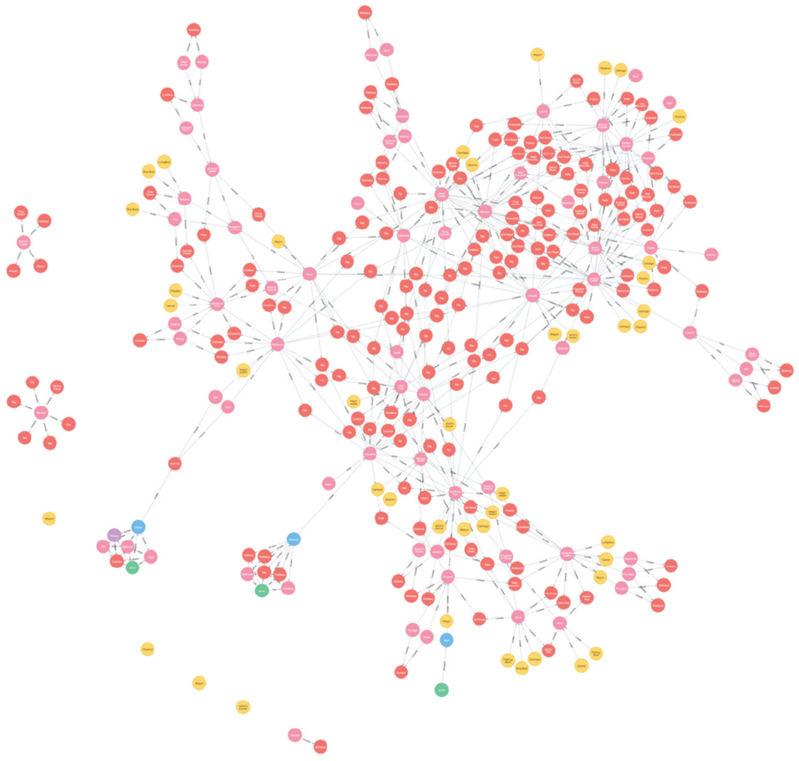


**Figure 3.** Sample story graph upon initialisation.

Graph can be considered a given character's collective understanding of its world. This graph only contains locations, assets, and characters that a character has seen and/or with which it has interacted before. This mechanism gives fAIble II the ability to tell what knowledge a character has about the world around it – i.e., its situational awareness. A sample story graph (note that each story would end up with a different graph upon initialisation) is shown in Figure 3. Although it is difficult to see the details, In this particular graph, the story takes places in space, thus many areas are connected through instances of 'sky'. Despite looking as identical copies, these nodes actually refer to different segments of space, allowing some to be safe, while others to be full of dangerous enemies or enticing wonders.

The Mind Graph is generated by a set of actions performed on a given character. The action inserts all nodes within one edge length of the character's location into the Mind Graph of the character. If a node already exists in the Mind Graph, the properties of that node will be updated. Upon creation of any nodes in the Mind Graph, new emotions are connected to the character based on the character's personality. Emotions and personality are the driving force behind character development; they allow the fAIble II system to make decisions based on a character's past experiences (indirectly), rather than solely on a character's attributes and intentions. Figure 4 shows an example of the Mind Graphs generated for the seven main characters in a story (blue nodes) within a sample story. Each mind will have its own set and interpretation of what it knows about the world and its inhabitants, allowing for different information and opinions, leading to different behaviours, even when given the same goals.

The Event Generation stage has fAIble II operating on the graph structure of the world that was created during the World Generation stage as described above. It does so in three phases: i) initialisation, ii) plot development, and iii) conclusion. All three phases make use of character agency,

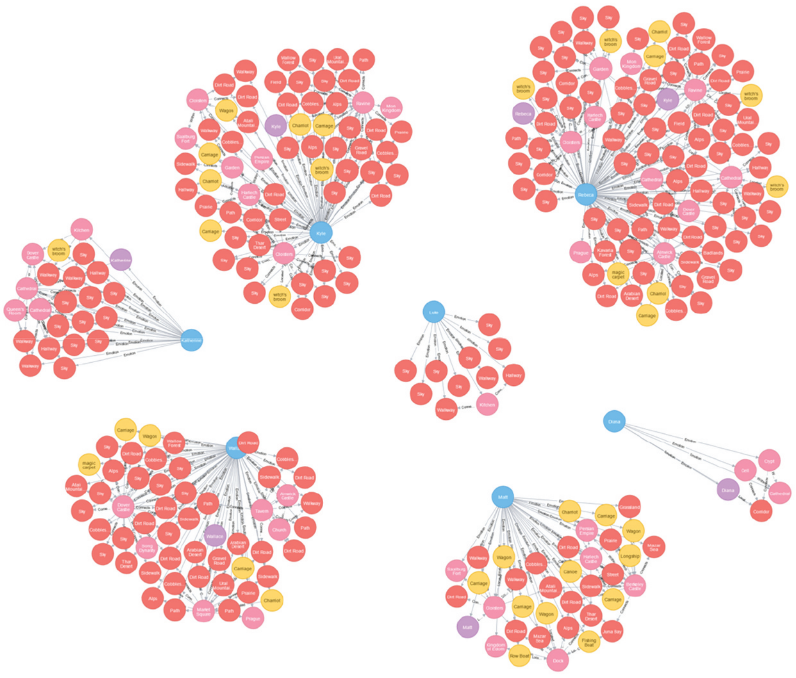


Figure 4. Mind graphs for all characters in a sample narrative upon initialisation.

a character's goals and the character's perceptions. Any character can be granted agency, as compared to fAIble I where only the protagonist has agency, and to AESOP, where no character had agency per se. Therefore, the characters act autonomously in the story.

When granted agency, a character becomes 'active' and selects the action to be taken next by it. The state of the story graph provides a finite set of actions that are available for the character with agency to take. Thus, action selection begins by collecting the possible actions that can be taken. Next the current state of the active character's Mind Graph is collected. Actions that are deemed not appropriate for the active character are filtered out (e.g., if Kahn is afraid of water then Khan would not want to chase Kirk into a river). Filtering criteria could be related to any element of the character's Mind Graph. This can include knowledge (or lack thereof) of certain parts of the world, the emotional state, and/or the current goals of the character. The possible actions left after the filtering represent sensible actions that the active character can take; the Event Generation system will then stochastically choose an action to perform from this set. Actions taken can change a character's emotions and personality – their own as well as that of other characters – as a way to develop the characters throughout the narrative.

As suggested above, fAIble II makes character development a central feature of the improvements made over fAIble I, specifically as related to emotions/personalities. fAIble II assigns personality and emotions to the characters. It uses Ekman's six basic emotions (Ekman & Friesen, 1978): joy, anger, surprise, disgust, sadness and fear, both for the emotions (sudden and short-term) and the personality (more stable and long-term) of a character. These supplement the 'physical and psychological traits' and 'attributes' of fAIble I and AESOP respectively. Unlike its two predecessor systems, fAIble II uses an active character's perceptions and emotions to influence the decision of what its next action will be. fAIble II also assigns roles, titles and gender to the characters, the last to ensure correct pronoun usage, as well as health statistics in real time.

## Natural language generation

fAIble II retained the use of simpleNLG as the basis of its NLG stage. However, it also introduced the Event Translator module that acts as an intermediate step between Event Generation and the invocation of the NLG engine. It parses the event representations individually as received from the Event Generator, and removes unnecessary information before passing them to NLG. The Event Translation step includes the following: i) framing the sentence by deciding whether adjectives are positioned before or after the verb; ii) providing phrasing for the emotion felt by the character to make the emotion more impactful; and iii) providing a generic sentence representation for the event. This sentence representation is composed of a JSON object that describes the event through the different components that make up sentences: noun, verb, adjective, adverb and prepositional phrases. From these individual components, the system can dynamically build two possible sentence structures: *simple* and *compound* sentences. A simple sentence can be defined as a clause with a single subject and verb (along with any adjective, adverb and prepositional modifiers), and that can stand on its own as a sentence. For example, 'The very charismatic and dashingly handsome captain deftly flew through the black hole'. A compound sentence is composed of two or more simple sentences joined with a coordinating conjunction. For example, 'The very charismatic and dashingly handsome caption deftly flew through the black hole and he saved the galaxy'. The decision to make a simple vs. compound sentence is decided by the Event Translator which has more contextual information so it can decide to combine events for compound sentences.

From these two sentence types, the system can provide a wide range of diversity that can be used in the narrative. Moreover, the fable II NLG system also adds additional variation by changing the composition of simple sentences through structural modification. An additional four sentence structures can be created based on the original sentence structure: a) sentence with front modifier, b) sentence with end modifier, c) sentence with front proposition; and d) inverted sentence. The sentence structure is randomly assigned during the sentence realisation process. The un-numbered table below demonstrates how these different sentence structures can change a sentence.

| Normal Sentence | a) Sentence with Front Modifier | b) Sentence with End Modifier | c) Sentence with Front Proposition | d) Inverted Sentence |
|---|---|---|---|---|
| *The wind suddenly rushed through the open window.* | *Suddenly, the wind rushed through the open window.* | *The wind rushed through the open window suddenly.* | *Through the open window, the wind rushed suddenly.* | *Through the open window, the wind suddenly rushed.* |

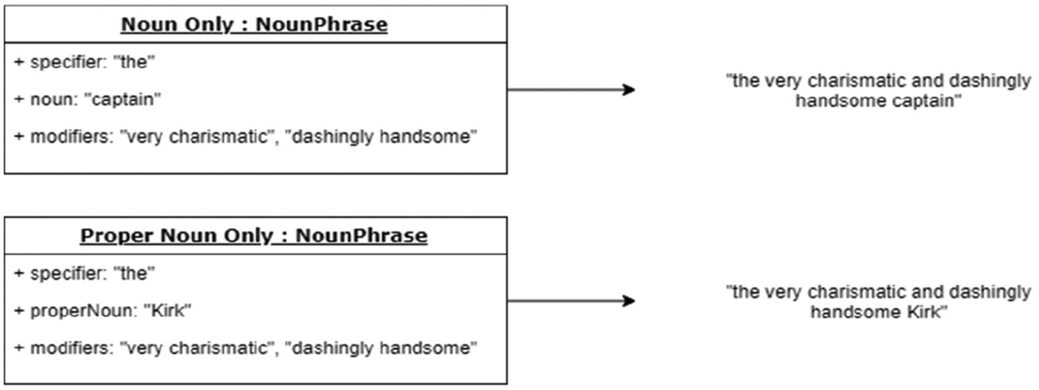

Noun Only : NounPhrase
+ specifier: "the"
+ noun: "captain"
+ modifiers: "very charismatic", "dashingly handsome"
"the very charismatic and dashingly handsome captain"
Proper Noun Only : NounPhrase
+ specifier: "the"
+ properNoun: "Kirk"
+ modifiers: "very charismatic", "dashingly handsome"
"the very charismatic and dashingly handsome Kirk"

Figure 5. Graphical depiction of how the NLG engine handles the inclusion of adverbs and modifiers.

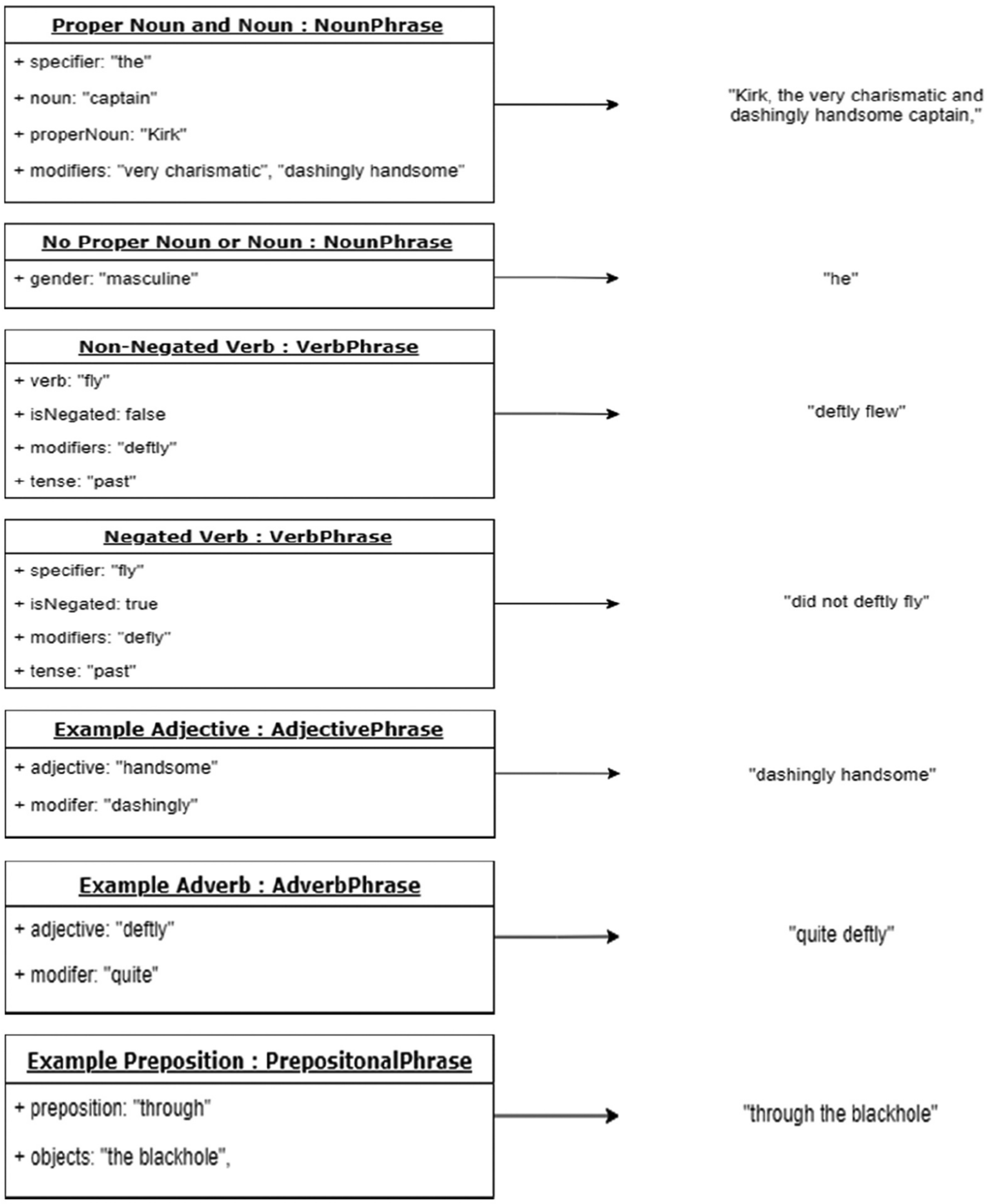


**Figure 5.** (Continued).

Figure 5 is a graphical depiction of how the NLG engine handles the inclusion of adverbs and modifiers. The depiction demonstrates how the different sentence components are structured and how different parameters can be set to change the components. For example, the noun of the sentences can be represented with as much or as little information as needed. This allows the system to transition from introducing a character, ‘Kirk, the captain’ to simply referring to the character by their designated gender, ‘he’. Verb phrases can also be specified to be negated or not to represent actions happening or not. The NLG engine also handles the inclusion of adverbs and adjectives as modifiers of noun and verb phrases, which gives additional flavour to the text. For example, the fAIble II NLG system can turn 'Kirk, the captain' into 'Kirk, the very charismatic and dashingly handsome captain', and 'flew' into 'deftly flew'.

In summary, the NLG component of fAIble II expanded upon the original concept from fAIble I by focusing on increasing the variation in the narrative text to avoid repetitive text. This was done by exploiting different types of sentences and structures. While the fAIble II NLG implementation accomplished this, it did have its faults. The random assignment of sentence structure can lead to clumsy and mechanical sounding sentences. Continued work on creating naturally flowing narratives was left to future iterations of fAIble.

Below is a story generated and narrated by fAIble II.

> Our story begins with Q, a civilian. He cared for Kirk. There was also Khan, a sailor. He took Kirk. Khan looked for a way to cross an antica bay. He saw a motor boat. He boarded the motor boat. He escaped from Q into the antica bay. Khan infuriated Q. He was on a quest to get Kirk back from Khan. Q, a civilian, looked for a way to travel across the antica bay. He noticed a yacht. He boarded the yacht. He went from a shuttle port to the antica bay. He crashed his yacht on his way to a shipping yard. Sailing worried him before but now terrified him. He noticed Spock. He made Q feel terrified. He decided not to ask Spock for help. Q saw Alice, a soldier. She made Q feel terrified. He decided not to ask Alice for help. Q looked for a way to travel across an indian ocean. He saw a motor boat. He boarded the motor boat. He went from the shipping yard to a rail way. He got off the motor boat. He looked for a way to navigate through a space. He was not able to travel that way. He saw Kyle. He made Q feel terrified. He decided not to ask Kyle for help. Q, a civilian, noticed a blizzard. The blizzard made him feel terrified. Though he was scared, he over-came the blizzard. He looked for a way to pass over the indian ocean. He noticed a motor boat. He boarded the motor boat. He escaped from the blizzard into the indian ocean. He proceeded to the shipping yard. He arrived at the shipping yard. He passed over to the shuttle port via the antica bay. He got off the motor boat. He looked for a way to go across a space. He noticed an archer. He boarded the archer. He went from the shuttle port to the rail way. He went from the rail way to a space. He crashed his archer on his way to a trade hub. Flying terrified him. He met Khan. He made Q feel terrified. He dropped Kirk. Khan fled. Q successfully got Kirk back.

While still not acceptable as fully interesting and coherent stories, it is clear that the story from fAIble II represents significant improvements from those of AESOP and fAIble I. First of all, the stories are longer and richer in detail and explanation. Secondly, the characters are endowed with some qualities as well as emotions. Third, there is a sense of geography and transportation that are generally appropriate. The main weaknesses are the lack of total coherence and the somewhat repetitive nature of the movements across the simulated world – seemingly going around in circles.

The reader will notice that the stories of fAIble II dispense with the thinking-acting feature that was so prevalent in fAIble I stories. While that was an innovative concept to have in the narratives and should be re-incorporated on a limited scale in a future version, constantly and awkwardly expressing thinking before every action became an annoying distraction. On the other hand, the narrative generated by fAIble II seems to overly focus on the characters moving around the world, and focus very little on what interesting and relevant other actions the characters should take. Ideally, these two aspects of a story should be weaved into the fabric of the story much more cohesively and synergistically. Nevertheless, there is an improvement in the quality of the natural language narration and the transitions between events seem to be generally smooth.

We refer the interested reader to (Alvarez et al., 2019) for a more detailed description of fAIble II. In the next section we next describe fAIble III, which is the final system created as part of our research.

## fAIble III – the final system

fAIble III (Bottoni et al., 2020) was the final system and the culmination of our work. The fAIble III system was built (philosophically, if not completely structurally) upon the work of AESOP, fAIble I, and fAIble II. The focus of this fourth and last cohort was on having the system build a more robust story world, one that included a backstory for each of the main characters. This would lead to explanations and justification for some actions taken by the characters, something that was missing in the first three versions. Modifications were also made to the event generation process so that stories could be more richly detailed and so that side stories take place in the course of the overall story. Our model for this feature was Homer's Odyssey, where Odysseus encounters several complex side adventures that temporarily interrupt his objective to return home to his family. Lastly, the NLG feature was improved by adding a function that looks for patterns in the text that can be replaced by a more streamlined and natural phrasing of sentences. More on this later below.

Similar to fAIble II, the fAIble III system is composed of the four main processes (World Creation, Event Generation, Event Translation and NLG); however, the processes work slightly differently. First, the story world is generated and the backgrounds of the main characters are created, giving the story a setting, and to the characters a history as well as connections to elements of the world. Next, events are generated based on a context-free grammar that allows for multiple sequential plot lines and subplots. These events are then passed to a translation module that converts the detailed event objects into sentence structures, and uses a pattern recognition function (mentioned briefly above and to be discussed in more detail below to diversify the sentence composition. Lastly, the sentence structures are joined into readable English sentences in the NLG module. We next discuss the fAIble III implementation of these processes.

## World creation: building a story world that addresses character depth

As a composer of children's fantasy stories, one major focus of the work on fAIble II was to explicitly define a world in which the characters live and act – who these characters are, what are their attributes, what locations they can inhabit, etc. Given that tales of fantasy generally ask the listener to suspend belief, it is important that the world be well-defined, especially if it is to include things like magic, mythical creatures and mysterious places. However, humans don't usually act without a reason, and characters in a story shouldn't either if the story is to sound logical and internally consistent. So, fAIble III built upon the World Creation stage introduced in fAIble II and extended it by creating the ability to explain character actions in the story, possibly because of events that took place before the story begins. This became one of the goals of the fAIble III work vis-à-vis world creation. Therefore, the purpose of the World Creation stage should not be to create a readable narrative – the generation of the main story does that. Rather, it should be to create a scaffold to support, justify and explain a character's relationships and actions in the main story.

As a result, a process was added to fAIble III to generate a background for the main characters that determines relationships that the characters will have had with other characters and items. The world creation step of fAIble II is otherwise very similar to that of fAIble II.

Inspired by the NPCAgency (Pickett et al., 2015) and Caves of Qud (Grinblat & Bucklew, 2017) systems/game platforms, fAIble III generates a 'history' – a *backstory* – for its characters before the story even begins. Backstory creation is an additional step added within the fAIble III World Creation module. As in fAIble II, characters are generated and put in the world, and one is chosen to be a protagonist. fAIble III adds the final step to create the protagonist's backstory after the rest of the world has been generated, but before the World Creation stage exits. Not only does this allow fAIble III to explain character actions, it also provides a way to lengthen stories by creating new events that instigate sub-plots with goals based on characters' personalities, emotions, and relationships. We refer to these as *plot hooks*. Plot hooks are pre-generated as part of the backstory sequence and serve to begin a new 'chapter' in the story. If the current plot line has concluded, the event generation algorithm will choose from the available plot hooks in order to create a new problem for the protagonist to solve.

Backstories are represented in the graph database, along with nearly everything else in fAIble III. The graph database has two parts as it relates to the backstory. The first is a relationship between the root node of the character's mind sub-graph and the node that represents the item or person to which the backstory is related within the mind sub-graph. The second is a node referred to as the *history* node in the graph's schema that serves to hold additional information about the backstory event. It is attached to the item or person's node within the character's mind sub-graph, and contains a description of the event that led to that relationship. All inter-character relationships, gained items, roles, etc. are stored as in the sub-graph that connects to the character. For example, when choosing a plot hook to use to start a new sub-plot, the system can query about other characters with which the main character has anv: `enemies` relationship, and then queries for a history node that contains the details about the event that caused them to be enemies. The backstory, therefore, explains all the connections and events in the graph. The edges of the nodes in the graph database have labels that describe what happened, so that if there is an event where an axe was gifted to a character, there will be two nodes stored within the character's mind sub-graph: 'Axe' and 'History', which are connected to each other and the root node of the character's mind, as follows: 'Mind' ←:gift ← 'Axe' ←:affects ← 'History'.

The fAIble II world creation system could generate sentences such as 'Chris valued a sword' and 'Marty stole his sword'; however, there was no explanation as to why Chris valued that particular sword, or why Marty would want to steal it. With the addition of the backstory feature of fAIble III, those sentences would sound more explanatory, such as 'Chris valued a sword because it was a gift from his father' and 'Marty hated Chris because they got in a fight. He stole Chris's sword.' These sentences still sound somewhat choppy, but they provide information about the characters' motivations, creating a more logical storyline. While the above examples may contain less than complete explanations, this fAIble III process could be easily augmented to provide a more extensive explanation.

The backstory generation module spins up the life story for the protagonist during the world creation stage of fAIble III (before the main story plot begins to unfold). This module could be used to generate a backstory for any other character besides the protagonist, but this is not incorporated in the prototype, where the use of the backstory module is to provide plot hooks for when the protagonist starts a plot; therefore, all of the backstory events created are centred on the protagonist. That does have the effect of generating a small amount of backstory for the other main characters, though, as the other important characters who make appearances in the main story are the same ones with which the protagonist has some sort of connection through the backstory.

A character's backstory life is abstracted as a progression of discrete events composed of three life stages: *youth, teenager* and *adult*. The youth life stage assigns the character birth parents and a place of birth. Characters must live in their place of birth for their entire youth stage. During this stage, characters are also required to gain a connection with another character (i.e., make a friend) and acquire an item (but not weapons). During the teenage life stage, characters must gain a connection with another character and acquire a weapon (either as a gift from someone, or by finding it somewhere, such in a cave or in a forest), as well as lose a weapon or other valuable item. The adult stage concludes the backstory generation process, and always involves two gained connections to other characters, and the acquisition of at least one more weapon or tool. Non-gifted weapons can only be acquired in the teenage and adult stages. Children cannot 'find' or otherwise acquire weapons, but someone can gift them a weapon (e.g., a family heirloom sword) for their use later in life.

Characters are assigned a role in life (e.g., pilot, teacher, farmer, baker, warrior, blacksmith) during a process that is separate from the development of the backstory. Roles are partly responsible for a character's attributes and weights such as strength or dexterity. These weights are a combination of a randomly generated base number plus a bonus dictated by their assigned role. Roles are determined during an earlier stage of world creation, when characters are first created and put into the world. All characters have roles, not just those which have gone through backstory generation. Additionally, at least one random event is chosen from the list of possible events for each stage of a character's life.

The generation of backstories for the characters requires that events be described, much like the events in the main story. However, these backstory events took place before the main story begins, and serve as a history lesson about the characters. These events include such things as gaining connections, weapons, tools, roles, relationships, etc. Each life stage contains a pool of possible events from which the system can select and add to a character's life. The events that actually take place in the backstory are selected: partly stochastically, but only loosely similar to how event generation happens in the generation of the main story because the backstory is divided into the three life stages and some events in each stage are mandatory while others are selected randomly. A set of possible events is based on pre-conditions and post-conditions assigned to an event, just like the contexts did for AESOP; for example, in order to lose an item, a character must first have had possession of it.

Each life stage, therefore, has one or more events that must happen, and some events that are not allowed. One important aspect of this process is that for each of the events that create relationships, whether with a friend, a family member, or an enemy, the character with which the connection is formed must already be a part of the story world; this way, these characters can make an appearance later to possibly influence the narrative.

The backstory influences event generation by generating plot hooks that are used to determine what plot-inciting events can occur. The event chosen by the Event Generator (see the next section) to be the next action to take place is determined by what available backstory there is from which to pick. For example, if a character was gifted a sword, then one of the events of the story might be that a villain stole the character's sword and the character who lost the sword now initiates a quest to recover it. When a new plot is being chosen, such as at the beginning of the story or when the protagonist accomplishes its goal and the story needs to move on elsewhere, the event generation module looks at a list of available plot types and narrows it down to those where the protagonist has the appropriate types of plot hooks, then randomly chooses one. For example, the plot of all stories generated by fAIble II, where an antagonist steals an item or kidnaps a friend and the protagonist goes on a quest to get it/them back, can be used in fAIble III if the protagonist has an enemy and either a friend or a valued item when it is time to start a new plot.

The backstory and event generation impact NLG indirectly – they just provide more sentences for the NLG to generate. The events passed to NLG have an optional 'reason' field to allow a justification associated with an event, such as valuing an item because it was a gift, to be passed to NLG.

The algorithm for the World Creation Stage is shown below.

## *World creation stage algorithm*

```
(1) Generate a unique token to identify the world being created
(2) Randomly select one of the available schemas (premade)
(3) Generate locations:
    (a) Retrieve the available locations from the chosen schema
    (b) Create a node representing the root location in the schema
    (c) Set location = root node
    (d) create(location)
        i. If location does not have possible child locations (specified in schema):
            1 Return
        ii. Randomly select 0+ (2+ in fAIble III) of the possible child locations specified in the schema
        iii. Create nodes representing the child locations, with a :within relationship to the current
             node
        iv. If the current location is a concrete location a character can enter:
            1 Connect the current location and all of its child locations (e.g. rooms) with transi-
              tions that allow characters to walk between them.
        v. For each child node:
            1 create(child)
        vi. Connect descendent nodes that have matching modes of transportation with transitions
            that allow movement between them
        vii. Return
```

```
(4) Generate vehicles:
    (a) Retrieve the available vehicles from the schema
    (b) Select a number n of vehicles to generate (in fAIble II and III, this is a configurable value)
    (c) While n > 0:
        i. Randomly select a vehicle type from the schema
        ii. Select a random location connected to a transition of the type the vehicle traverses
        iii. Select a name for the vehicle from the aliases available in the schema
        iv. Create a node representing the vehicle at the chosen location
        v. fAIble III only – create a second copy of the vehicle in the same way
        vi. n–
(5) Generate characters:
    (a) Retrieve the character information from the schema
    (b) Select a character to be the protagonist and one to be an antagonist (unused in fAIble III, but
        still generated as an artifact of fAIble II)
    (c) For all other characters in the schema:
        i. Create a node representing the character at a random location
        ii. Create the root of the character's mind sub-graph
        iii. Randomly select n between 4 and 10
        iv. While n > 0
            1 Move the character to an adjacent location
            2 Add everything in the location to the character's mind
            3 n–
(6) Generate assets:
    (a) Retrieve information for all available assets except vehicles from the schema
    (b) For each type of asset:
        i. Select a number n to generate (configurable in fAIble II and III)
        ii. While n > 0:
            1 Randomly choose one asset of the relevant type from the schema
            2 Create a node representing the item
            3 Place the node at a random location
            4 n–
(7) Generate quest:
    (a) In fAIble II:
        i. Choose an item or person to be the target of the quest
        ii. Place the protagonist and antagonist at the same location as the target
        iii. Create a quest to retrieve the item within the character's mind
    (b) In fAIble III:
        i. Place the protagonist at a random start location
(8) Generate backstory for the protagonist (fAIble III only):
    (a) Youth stage:
        i. Select two other characters at random to be the character's parents
        ii. Set the character's current location as their birth location
        iii. Give the character a connection and a tool
        iv. Randomly select whether to give the character an additional connection or tool
    (b) Teenager stage:
        i. Give the character a weapon and a connection
        ii. Choose one of the character's connections at random to lose (e.g. death, fight)
        iii. Randomly choose one of the following:
```

1 *Gain a connection*
2 *Gain a tool*
3 *Gain a weapon*
4 *Lose a connection*

(c) *Adult stage*
  i. *Give the character two connections and a weapon*
  ii. *Randomly choose one of the following*:
    1 *Gain a connection*
    2 *Gain a tool*
    3 *Gain a weapon*
    4 *Lose a connection*

## *Event generation*

The next step of the fAIble III system is event generation, which weaves together the plot of the story. One important element of a story for enjoyment by children (and adults too!) is that some of the actions taken by characters be surprising. Therefore, a traditional AI planner such as used in many other reported works in automated story generation was not deemed a good match for our work.

In fAIble III, event generation is a mix of stochastic selection of actions and pre-scripted selections. The fAIble III event generation works by dividing the story into stages, which determine what events are available and what the character's immediate goal is. This process follows an algorithm loosely modelled after a replacement grammar, shown below:

$$
\begin{aligned}
&S \rightarrow iPo\\
&i \rightarrow intro\\
&o \rightarrow outro\\
&P \rightarrow eG \mid G\\
&G \rightarrow fTR \mid TR \mid R\\
&f \rightarrow searchforinformation\\
&T \rightarrow eT \mid PT \mid t\\
&t \rightarrow travel\\
&R \rightarrow r \mid rP\\
&r \rightarrow resolvecurrentplot\\
&e \rightarrow \text{good } event \mid \text{bad } event
\end{aligned}
$$

In practice, what this means is that the event generation module keeps track of the current stage in which the plot being generated is and emits pre-scripted sequences of events when appropriate (e.g., when starting a new plot or resolving a finished one), determines the events available to the character, and determines which random events, if any, can occur in the given stage (e.g., *interrupts* are described later).

In fAIble III, a complete narrative (S) consists of an *intro*, a plot (P), and an *outro*. These are a significantly more structured versions of the *initialisation*, *plot development* and *conclusion* phases of the fAIble I Event Generation stage. The intro and outro are pre-written by the system, and every narrative contains exactly one of each. The intro is pre-written right after the world creation stage. The main character's name, its role, and some other story details are woven in to provide an introduction to the main character, to the antagonist, and to the setting of the story. A set of possible outro's is also pre-written – even before the main narrative itself is generated by the system. This is possible because the possible outro's are few in number and they are quite general. The appropriate one is selected based on the schema of the story, in spite of the fact that the main story narrative has not yet been written by the system. The intro and outro are stored as nearly-complete sentences. This is possible because of the finite number of possible plots in the story world, it is reasonable to have these pre-written by the system. The outro fills in the necessary details for a satisfactory ending. These serve as bookends to the story, so that even if the events within the story appear random and chaotic, there will at least be a definite beginning and ending to the story.

The plot – the meat of the story – consists of *good events* and/or *bad events* happening to the protagonist (e) while it pursues goal G – the process of the protagonist pursuing a goal. A plot can be the main plot that starts after the intro, a subplot that occurs when the protagonist is travelling, such as for example, needing to find a key to continue, or a new plot that begins after the previous one is completed. Pursuit of a goal, as shown in the grammar, can have several different structures but usually follows the order 'fTR,' that is, a search for information (f), followed by a travel sequence (T), ending with the resolution of the current plot/goal (R). Within travel (T) and resolution (R), there exist the possible structures 'PT' and 'rP,' respectively. As 'P' represents the creation of a new goal, these replacements present the opportunity to branch into a new subplot. However, it is possible for a story to complete after the pursuit and resolution of a single goal and avoid branching altogether. Thus, during travel, sub-events can occur, a subplot can be started, or the main character can simply travel unimpeded to its destination. During the resolution, the current plot is resolved, after which a new plot can be started by a plot hook as described next. Basically, the event generation algorithm uses the backstory events to help determine what events are possible and to create the plot hooks. Because a new plot can start during travel (by expanding T -> PT) or after resolving the current plot (R -> rP), this permits having multiple plots and subplots in the story.

A character's backstory and previous story events allow for the creation of plot hooks. These are the beginning of new sub-plots at various places throughout the narrative, where appropriate. For example, if the main character goes on a quest to obtain a magical sword about which it had heard legends, a sub-plot where the antagonist steals the sword again right after the protagonist finally acquires it could be a possible topic for continuing the story. This plot hook feature of fAIble III not only allows for richer (and longer) stories, but also makes story progression more logical. To clarify, backstory events are used to determine which plot hooks are available and where sub-plots could be started in the story. If the character has a friend and an enemy, then a plot where the enemy kidnaps the friend is an option available when the story begins. The event generation module chooses which inciting event occurs from the ones made possible by the character's backstory.

The replacement grammar could be thought of as splitting the story into stages where different types of events are available. During a plot introduction, the events are largely scripted, but during the information gathering stage the character wanders using the same stochastic mechanism as fAIble II to select the appropriate action. After getting the desired information, the character travels to a destination along the shortest known path, but randomly generated events such as *interrupts* can occur along the way.

Interrupts are a concept that was carried over from fAIble II but was redesigned to fit in the revised plot generation system. Interrupts are events that have a certain probability of happening between every event in the story. They usually provide some challenge to the protagonist and are intended to make the story a little more interesting. In fAIble II, interrupts were events that could occur randomly to the main character, and consisted of, for example, a storm forcing the main character to escape from a location, or a vehicle being used by the main character breaks down or crashes. In fAIble III, the available types of interrupts are a storm from which the character is forced to escape, and an enemy (such as a wild animal or a bandit) that attacks the main character. The vehicle crash interrupt type was removed because it frequently led to the story becoming uninteresting when the main character would crash in an area from which it could not escape without the now-broken vehicle. Of course, these dead ends could be resolved through the deus ex machina feature originally introduced in fAIble I. The current fAIble III prototype does include the ability to perform deus ex machina events, but its use was purposely minimised to avoid it becoming too easy a solution and therefore become overused.

These interrupts are implemented as sub-plots that go through all the same stages as the story's main plot – they are just shorter because the character starts off very close to the goal location and with the character already knowing the path to the goal location, meaning there is no need for information gathering or travel stages. An example of an interrupt could be a storm suddenly appearing while the protagonist is trying to sail across an ocean. The 'resolution' for these plots involves an escape from the storm in the storm example, or either an escape from or defeat an enemy in the other type of interrupt. An interrupt can occur at random any time when the main character is travelling towards a goal; which interrupt is chosen is also random.

Event generation is split into multiple stages modelled after the replacement grammar described above that is intended to represent the structure of stories in methodologies such as Kurt Vonnegut's 'Shapes of Stories.' Much as in fAIble II, the events are generated through a mixture of stochastic and pre-scripted processes. When a plot is beginning, a plot type such as 'fetch quest' or 'recovery' is chosen at random from those which have their plot hook requirements satisfied by the main character's backstory or by previous events in the main story. The introductory scene for each plot is largely pre-scripted, but it is then followed by a stage in which the character explores, seeking out information about where to go. Once the character knows where it should go to achieve its goal, it will follow the shortest path to it; however, random events such as interrupts can occur along the way, or an entirely new sub-plot may need to be started if the character needs to go find a key for a door to continue. Most of the logic for determining the next event within a stage is taken directly from fAIble II – e.g., the decision of which direction to wander, the way the character determines the shortest available path to a goal, and the way a character decides whether to use a vehicle to traverse from one area to another all remain unchanged. This splitting of the story into stages with the replacement grammar model removes the statelessness inherent with fAIble II, and instead allows for more structured stories that can accommodate more than one goal for the protagonist and allow chaining multiple plots together.

Plot hooks are stored in the main character's mind sub-graph as a relationship between the root of the character's mind and another entity of which the character is aware, such as an item or person, that has a label indicating what type of plot hook it is (e.g., a friend that the main character has, who could potentially be kidnapped), and as a node (referred to as another 'history' node) that has further information about the plot hook, so as to allow the system to provide a reason for a character's actions. Each plot type, which determines the inciting events, goals, and resolution of a given plot, has certain plot hooks that are prerequisites for it. Any time a new plot needs to be started (for example, at the beginning of the story or after the main plot has been resolved), the system queries the character's mind for existing plot hooks, determines which plot types have their prerequisites satisfied by the available plot hooks, and then chooses at random from those. This allows the story to tie into the character's backstory (by explaining why a character valued an item enough to go on a quest to get it back, for example) and to provide some continuity between parts of stories. Because events that occur in the story, such as going on a quest to find an item, can add to the available plot hooks, it is possible for the story to use past events as the basis for a new plot, such as having an item that the character had previously gone on a quest to find, become stolen by an enemy.

The algorithm for the Event Generation stage of fAIble III is described next.

## *Event generation stage algorithm*

```
(1) Generate event for introduction (pre-written)
(2) Generate a plot for the main story:
    (a) While the length(story) < minimum length (arbitrary, but set to 100 events in fAIble III):HT
        i. Query the protagonist's mind for plot hooks
```

```
          ii. Retrieve the list of available plot types
         iii. For each plot type:
                  1. If the plot type's requirements are met by the character's plot hooks:
                        a. Add to a list of eligible plot types
          iv. Randomly select one from list of eligible plot types
           v. Push a goal onto the stack of goals in the protagonist's mind reflecting the goal of this
              plot type (e.g. to retrieve a stolen item)
          vi. Generate pre-scripted inciting events for the plot type (e.g. enemy steals item)
         vii. If the protagonist is not already at the location of the plot's goal:
        viii. Push a goal to travel to the main goal onto the protagonist's goal stack
          ix. Push a goal to locate the main goal onto the protagonist's goal stack
           x. While the protagonist does not know of a path to the main goal:
                1 Check the protagonist's location for items
                    a If there is a key, medicine, tool, or a weapon better than what the character has,
                      pick it up
                2 Check the location for other characters to ask about the goal
                    a If the protagonist hates or fears the other character, skip asking
                    b Else, ask if they know how to get to the goal
                        i If they do know, update the character's mind and break
                3 Choose an adjacent location that the protagonist can traverse to
                4 Move the character to the adjacent location and generate a "wander" event
                5 Update the protagonist's mind by adding everything in the current location to it
                6 If the character has been wandering too long (>15 locations in fAIble III), generate a
                  map to the goal for the character to find so that the story does not get boring
          xi. Pop the locate goal from the stack
         xii. While the protagonist is not at the location of the main goal:
                1 Attempt to travel one step along the shortest path in the character's mind between
                  the current location and the main goal
                2 If the character was unable to travel:
                    a Recursively start a new plot, using the reason the character was unable to travel
                      (needing a key or needing a vehicle) as the only available plot hook
                3 Else, determine whether to spawn an interrupt at random.
                    a If yes, recursively start a new plot, with the available interrupt types as the only
                      available plot hooks
                4 Generate an event representing the movement (e.g. walk, drive, fly)
        xiii. Pop the travel goal from the stack
         xiv. Generate the scripted events for the plot resolution
          xv. Update plot hooks based on events of the plot (e.g. if the character went on a quest for
              an item, it can be used as a plot hook as an item that can be stolen)

  3   Generate scripted events for outro
```

## Character development in fAIble III

There isn't an explicit module or piece of code that deals with character development. fAIble III uses the 'emotions and personalities' approach introduced in fAIble II nearly unchanged, where characters' emotions (e.g., love, hate, fear) towards other characters or towards things can be changed throughout the story. This is usually based on an event, such as when a character saves another character's life, which would move the first character's emotions away from 'hate' and towards 'love'; as in fAIble II, there is also a chance for random changes in character personality throughout the course of the story, although the probability of this is small.

## *Natural language generation in fAIble III*

In the context of our project, fAIble I was the pioneer in creating the NLG module for our system, and its basic foundation remained largely unchanged in fAIble II and III; however, some improvements were made by each of the subsequent systems. fAIble II added the Event Translation step as a separate stage to the NLG module to better structure the sentences and increase structure diversity. This Event Translation module was enhanced by fAIble III to incorporate the pattern recognition function that was first mentioned above that helps to further reduce unnatural repetition.

More specifically, the pattern recognition feature of fAIble II seeks to reduce repetition in the narration text and improve the overall quality of the sentence by analysing how sentences are processed in the Event Translation step (prior to reaching the NLG module). The fAIble II event translation process decomposes a story object into its scenes; it then passes each scene to a *generate* function that splits up the scene into its individual events and processes these events one by one. There is a function for each event; however, in fAIble II these functions were invoked in nearly complete isolation from one another. This resulted in repetition and overlooked opportunities to correct the repetition because sentences in stories are affected by the sentences around them, and most readers can quickly notice if they don't flow together 'correctly': e.g., 'Noah went to the store. He went to the store to buy eggs.' or 'Marie saw a spaceship. She boarded the spaceship.' We place quotation marks around 'correctly' because prescriptively, these sentences have correct grammar. However, they sound robotic and not human – 'incorrect' in a rhetorical sense. The core issue with those sets of sentences is of course, the pattern of repetition. Most of the time, readers don't expect to hear the same noun-verb pair or direct object in two consecutive sentences. However, in reality it is impossible to correct these kinds of contextual problems without having any information about the surrounding sentences. Therefore, the Event Translation system was modified with this pattern recognition feature to allow it to process multiple events at once.

The pattern recognition function addresses this isolation to achieve its goal of reduced repetition. It works by creating a second, associated function for each of the action functions. These output a clause object instead of a whole sentence object; these clauses could be then used as building blocks for more complex sentences. The information contained in a clause object and in a sentence object are nearly identical, but sentence objects are standalone. Because of the structure of the system, multiple clause objects can be combined to make one sentence. Because the action functions were no longer returning entire sentences, mappers were created inside the actions that were the beginning of a common sequence. Improvements made in fAIble III now allow such patterns to be recognised and combined into one or more sentences, while individual actions that were not part of a pattern are processed separately. So, instead of stating 'Nico noticed the pirate ship. He boarded the pirate ship.' the 'notice' – 'board' pattern would be recognised and thereafter combined into 'Nico noticed and boarded the pirate ship.' Similarly, a pattern would be condensed from 'Ghita saw a sword. She picked up the sword.' to 'Ghita saw a sword and picked it up.' All of the action objects passed to the NLG module are pre-processed in this way by the fAIble III pattern recognition function, allowing for more fluid paragraph structures and decreased repetitiveness between sentences.

The NLG module itself in fAIble III is unchanged from that of fAIble II. The material changes made in fAIble III vis-à-vis NLG took place in the Event Translator stage. This algorithm is shown next.

## *Event translation stage algorithm*

(1) *Receive 'story' object from event generation stage. The story object contains all the story events generated in chronological order.*
(2) *Create an array to hold the processed 'scenes' (combinations of one or more events)*
(3) *Set the story's tense to 'past'.*

```
(4) Begin the 'tick' system, which keeps track of "when" events happened (necessary for building
    understandable sentences)
(5) Until last event in the story object is processed:
    (a) Pass events array to the appropriate event function
    (b) If there is(are) pattern(s) starting with this event (as found in the event function), check for
        those pattern(s)
    (c) Verify that the subjects and/or direct objects and/or indirect objects (whichever are necessary
        for that particular pattern) match between the events
        (i) If they do match, then:
            (1) Process that pattern as a scene, and
            (2) Create individual event phrases that are bundled into a clause and returned as
                a scene to be added to the story. Scenes can contain one or more sentences
        (ii) If they do not match, or if there are no possible patterns starting with the current
             event, then:
            (1) Process the event by itself and return it as a scene
    (d) Remove any unnecessary fields from the events
    (e) Format the events in a form readable by the SimpleNLG system in the NLG module.
    (f) Remove the events that are used in the creation of the scene from the events array.
    (g) Add the scene that is returned to the scenes array
(6) Once all events are processed, pass the scenes array to the NLG module
```

The story object is received by the Event Translator directly from the Event Generator after all the events that encompass the entire story have been generated. It contains all these events in an *events array* where each event is sequentially placed in increasing index numbers. A second array called the *scenes array* is created by the Event Translator where the processed scenes are added. A scene can consist of one or more events. The story's tense is set to a default of past. While other tenses could be selected instead, that would require some manual action on the part of the author; in reality, it is unlikely that this will be necessary for the great majority of stories.

The *tick system* assigns relative times to every event so that understandable English sentences can be generated in the correct temporal sequence of events. If a sentence references another event, the algorithm can check the ticks of the two events to see in what order they occurred. Step 5a in the algorithm above selects the appropriate function that has been linked to the type of event being processed. Such linkages are done a priori by the system.

In order to demonstrate the result of the changes made to the fAIble III system, we implemented them on the introductory parts of the fAIble II stories. This way, we are able to juxtapose introductions to the stories generated by both versions of fAIble, and see how these modifications enhanced the system output. Below are examples of a fAIble II introduction and a fAIble III introduction for comparison:

fAIble II:

Story 1: *Once upon a time there was John, a civilian. He valued a sword. There was also Khan, a sailor. He took John's sword.*

Story 2: *Our story begins with Q, a pilot. He cared for Kirk. There was also Thomas, a knight. He took Kirk.*

fAIble III:

Story 1: *Once upon a time there was Martin, a prince. He valued a war hammer because it was a gift. There was also Anthony, a doctor. Martin disliked Anthony because they were lifelong rivals. Anthony took Martin's war hammer.*

Story 2: *Our story begins with Peter, a soldier. He cared for Kirk because they grew up together. There was also Alex, a knight. Peter hated Alex because they were friends but got in a fight. Alex kidnapped Kirk.*

The syntactic complexity has increased in the last two introductions. It still sounds somewhat telegraphic, but the additional subordinating clauses add to both the readability and the logic of the story. The further explanation of character actions leads to greater character depth.

An important part of stories is allusion, or indirect references to past events. An introduction such as this one generated by the fAIble II system …

> Our story begins with Luke, a mercenary. He cherished an iron hammer. There was also Christine, a queen. She took the iron hammer. She looked for a way to pass over a thar desert. She saw a carriage. She boarded the carriage. She got away from Luke, a mercenary, into the thar desert. Christine infuriated Luke. He was on a quest to get the iron hammer back from Christine.

… would sound like this when generated by the fAIble III system:

> Our story begins with Gwen, a blacksmith. She valued a rake because she went on a quest to find it. There was also George, a prince. Gwen disliked George because they were friends but got in a fight. George took Gwen's rake. George escaped from Gwen into a walkway. She decided to go on a quest to get back from George.

While the fAIble II stories gave the reader all of the information about a situation up front, the fAIble III system is able to allude to a past quest and a fight between friends that was part of the characters' backstories. The background generation structure adds more complexity and internal logic to generated stories, as well as increased explanation of character reasoning behind its actions, leading to deeper and more believable characters.

Below is a full story from fAIble III.

> Once upon a time there was Martin, a prince. He valued a war hammer because it was a gift. There was also Anthony, a doctor. Martin disliked Anthony because they were lifelong rivals. Anthony took Martin's war hammer. Anthony got away from Martin into a dirt road. He decided to go on a quest to get it back from Anthony. Martin noticed Thomas. He scared Martin. He decided not to ask Thomas for help. Martin looked for a way to journey Wallow Forest. He saw a carriage. He boarded it. He went from a church to Wallow Forest. He wandered to a grotto. He arrived at the grotto. He found a herb. He picked it up. He got off it. He went from the grotto to a dirt road. He noticed Kate. Martin asked Kate, "Where is a war hammer?" She told Martin how to find it. He meandered to the grotto. He arrived at the grotto. He wandered to a sidewalk. He went from the sidewalk to a cathedral. He arrived at the cathedral. He met Anthony. Martin noticed Anthony. Martin swung at Anthony. He dodged Martin's attack. He swung at Anthony. Martin won the fight against Anthony. Although Martin was victorious, his adventure was not yet over. He cared about a war hammer because it was a gift. There was also Wallace, a princess. Martin disliked Wallace because they were friends but got in a fight. Wallace took Martin's war hammer. Wallace got away from Martin into a dirt road. He decided to go on a quest to get it back from Wallace. Martin went from the cathedral to a cobblestone path. He went from the cobblestone path to a seaport. He arrived at the seaport. He found a root. He picked it up. He found a fairy nectar. He picked it up. He went from the seaport to a cobblestone path. He meandered to a tavern. He arrived at the tavern. He found a herb. He got it. He found a chisel. He got it. He found a war hammer. He picked it up. He noticed Christine. Martin asked Christine, "Where is a war hammer?" She told Martin how to find it. He went from the tavern to a dirt road. He went from the dirt road to a seaport. He arrived at the seaport. He found a plow. He picked it up. He found a herb. He picked it up. He found a plow. He picked it up. He found Excalibur. He picked it up because it was better than his war hammer. He found a mace. He did not get it because it was worse than his Excalibur. He met Wallace. He apologized to Martin. Wallace gave it to Martin. He successfully got it back.

Of course, fAIble III is still far from perfect, as the tests that we cover later in this article indicate. However, fAIble III is the product of evolutionary improvements on the system going back to Campfire, and will be the base of our future research.

## Alternative user interfaces

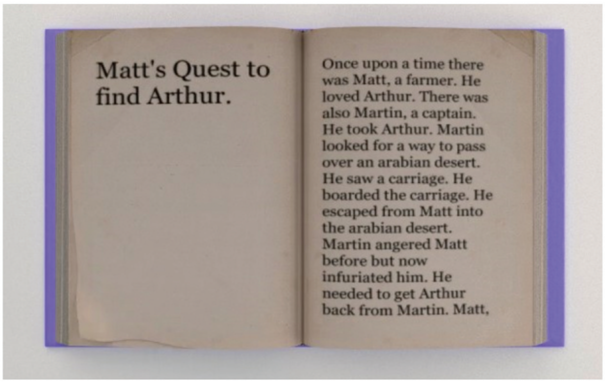
The output of the first two versions of the system (AESOP and fAIble I) was a simple block of text to be read to (or by) the listener. This was the obvious initial choice for the purposes of our research – to keep simple the peripheral aspects of the work and focus on what was most important at the time.

However, this is clearly not a good option for children who may not yet know how to read, or may be in the process of learning to read. Therefore, as side initiatives in our work for the last two versions (fAIble II and III), we explored alternative means of telling the story.

The first of such products was an open book-like graphic where the text of the story is contained in its open pages. See, Figure 6 above.

The story is written out over several pages of the book. Of course, the story has to be first created by fAIble, so the story has been already composed when the book feature is launched. Each page contains only a part of the story. After a few seconds to give enough time for an adult to read it comfortably, the pages gracefully turn automatically to expose the continuation of the story text. This process continues until the last page, where the story ends and the words 'The End' appear. While this output approach does not address the lack of reading skills of young users, it can serve to help those who might be learning to read if it is linked synchronously to a text-to-speech system (which we did not do, but would be easy to do).

As a second and more promising alternative output mode – one that does address children who do not yet know how to read – was inspired by the storyteller avatar used in Hollister's Campfire system. This output feature in Campfire uses a lifelike avatar (of Hollister himself) to relate the story out loud via a text-to-speech system while the storyteller avatar stands in front of a burning fireplace. See, Figure 7. Proprietary restrictions prevented us from using Campfire's avatar, so we developed our own avatar as part of fAIble III, which can likewise narrate the story out loud using a text-to-speech system. Figure 8 depicts our avatar developed as part this work.



Matt's Quest to find Arthur.

Once upon a time there was Matt, a farmer. He loved Arthur. There was also Martin, a captain. He took Arthur. Martin looked for a way to pass over an arabian desert. He saw a carriage. He boarded the carriage. He escaped from Matt into the arabian desert. Martin angered Matt before but now infuriated him. He needed to get Arthur back from Martin. Matt,

Figure 6. Active book on which the text is recorded.

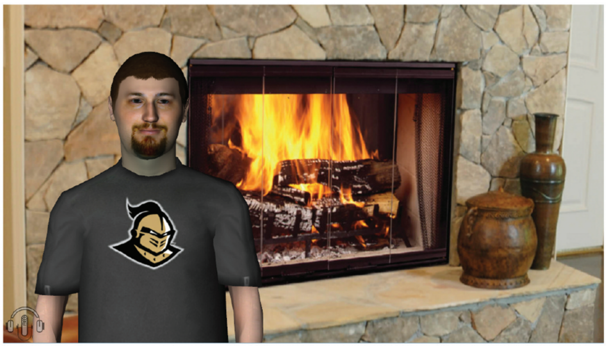


Figure 7. Campfire's storytelling Avatar.

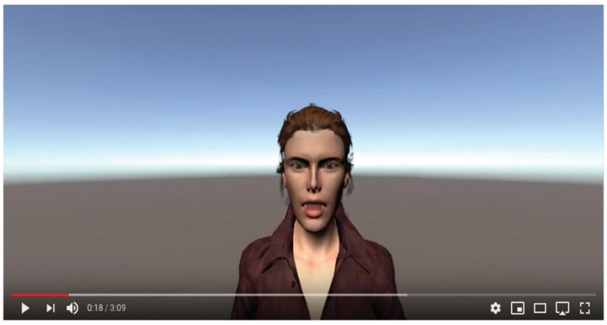


**Figure 8.** Storyteller Avatar depiction for fAIble III graphic interface.

While our avatar is not particularly good, nor is it endowed with a natural-sounding voice, a better version of such a narrator avatar could be integrated in future research to read the story to pre-school children, possibly as bed time stories. Admittedly, further research will be necessary on what would be the best appearance (e.g., a cartoon, or a lifelike avatar of the child's parents, grandparents or even of her/his pet) and the best voices for the avatar. The background scenery also needs additional work as to what would work best. It may be that the background would best be related to the theme of the story being told.

Finally, we should mention that we believe that a combination of the two output modes – the book and the avatar – to be particularly beneficial together for children learning to read, all in the context of a story. Furthermore, we plan to include in our future research an enhancement to the book feature that adds a live pen that writes the story text dynamically in real time as the words are being uttered by the avatar.

## Assessment of fAIble stories

Extensive and robust human subject testing of stories produced by the four prototypes was undertaken by each of the four cohorts on their respective prototypes independently. In addition, a final comparative evaluation of stories from all four prototypes at once was done to provide a direct comparison of the four versions under the same conditions. Moreover, a computational analysis of the readability of the stories generated by the four versions was carried out to complete the assessments of the research outputs. We should note here that although fAIble was targeted for children ages 6–8 years old, the test subjects used were all adults. Testing with children that young introduces a dimension of difficulty that was beyond the scope and capabilities of our computer science research group. An attempt was made in the comparative tests to request that those adults who had experience with children of that age group could answer the questions as they though the children would. However, very few participants answered in that manner, making the sample size much too small for a meaningful finding. Therefore, we do not include those results here.

### *Description of the independent human test subject assessments*

The bulk of the evaluations were done using human test subjects, as obtaining the opinions of the likely consumers of the produced stories was considered the most important assessment criterion. The procedures used for the three fAIble prototypes were similar to each other. They consisted of having a test subject read stories generated by the fAIble prototype being assessed, and then take a survey to answer some questions. The first six questions remained constant throughout all fAIble versions testing so that a direct comparison could be made later. Additional questions were included to learn more about the specific features of each particular version. The procedures used in the AESOP testing however, were quite different from those used for the three fAIble versions. We begin with AESOP.

## *AESOP assessment process*

The AESOP system used the rule sets in the six different modules discussed above to generate stories. Given the highly exploratory nature of AESOP, the objective of the evaluation was to shed light on which module would work best at creating the desired effects. Thus, the AESOP assessment involved generating 40 short stories (all of AESOP's stories were markedly short) as test cases. Seven test subjects were used in this evaluation. These test subjects were directly associated with the project – the four student developers, the PI and the two German mentors. The potentially biased nature of the test subjects and the small sample size invalidated these tests as scientifically meaningful. Nevertheless, they provided a modicum of feedback that we found to be useful at that stage of our research. The test subjects were asked to score each test case as follows:

- •• A story earned a rating score of **2** if the reviewer believed the story was coherent and consistent, and had a definable beginning and end.
- •• A core of **1** reflected a story that the reviewer found coherent and consistent with itself. Yet, it was not deemed optimal.
- •• Finally, a score of **0** meant that the reviewer found it inconsistent with itself and illogical. This rating was meant to represent stories which had no direction.

As a reminder, the six different modules described earlier in this paper are briefly described again as follows:

**Random** (5 test cases): The sections of the prototype code that filtered actions deemed impossible or improbable were disabled.

**Possible Actions** (5 test cases): Random, but Impossible actions are filtered out.

**Character Attributes** (5 test cases): Action selection is weighted towards character Attributes.

**Character Relationship** (10 test cases): Action selection is weighted towards character relationships.

**Character Goals** (5 test cases): Actions that create the Conditions that lead to characters' wishes becoming true are more likely to be selected. In these test stories, the goals assigned were: Gary wants to fight Harriette; the Witch loves Gary; and Harriette wants to kill the Witch.

**Context** (10 test cases): The context module incorporates a progression of author goals, ensuring that certain Actions occur in a specific order – but not necessarily linearly – during the course of a story. Unlike other stories that were terminated after 20 events, these stories terminate when the storyboard is complete. In the *Revenge* storyboard (5 test cases), first Harriette agrees to marry the witch, then Gary kills the witch, and finally Harriette kills Gary. In the Change-of-Heart storyboard (5 test cases), Harriette and Gary first fight, then become friends, and finally join forces to kill the Witch.

The results are shown below.

## *The fAIble assessment process*

The assessment process designed for the three fAIble systems involved having the fAIble prototype generate two stories. The human test subjects, who were NOT associated with the project in any way, reviewed the two stories and completed an anonymous online questionnaire that allowed them to rate the stories in some categories of desired qualities. As mentioned earlier, the questionnaire (survey) contained the same questions #1 through #6 for each of the three assessments (one for each version of fAIble). One thing that did vary was the makeup of the test subjects. For fAIble I they were recruited anonymously through the Internet; for fAIble II, students in a specific communication course at UCF were given access by their instructor to perform the evaluation. For fAIble III, a combination of the above two sources was used. Nevertheless, all responses were collected via the same anonymous survey hosted online by a third party service. The human test protocols were approved by the Institutional Review Board at the University of Central Florida. No written consent was required. The sample sizes varied during the succeeding versions: 41 for fAIble I; 75 for fAIble II; and 61 for fAIble III.

The first six questions were scored out of a maximum possible points using the grading scale: **2** for ‘Yes’, **1** for ‘Somewhat’ and **0** for ‘No’. A question could score a maximum number of possible points if all test subjects answered ‘Yes’ (2 points). Thus, for fAIble I, the maximum number of points per question was $41 \times 2 = 82$ points; for fAIble II it was $75 \times 2 = 150$ pts; and for fAIble III $61 \times 2 = 122$ pts.

The six common questions in the surveys were the following:

1. *Do story events appear to follow a coherent progression?*
2. *Do the characters appear to act based on some internal reasoning and motivations?*
3. *Do story events appear varied?*
4. *Does the language resemble human generated narrative?*
5. *Does the use of adverbs and adjectives add to the depth of descriptiveness of the story?*
6. *Is the language varied across sentences and stories?*

An example of other optional questions asked that were specific to each version is: (for fAIble II and III only)

7. *Does the animation enhance the experience of the story?*

Lastly, three open-ended questions were posed, asking the subject to name the system’s weaknesses, strengths and any additional comments.

Each version of fAIble (including AESOP) was assessed independently of the others, and at a different time, about one year apart. These sets of tests did not in any way involve the prior tests done on previous versions of the prototype, but rather, were self-contained and independent of each other. The only commonality was the first six questions of the fAIble surveys (but not AESOP). We next provide the results obtained for each of the four versions of the prototype. The results are shown below.

## *Results of the independent human test subject assessments*

First we briefly discuss the results for AESOP. Figure 9 depicts the average scores assigned to the sets of stories generated with the same rule sets. From these results, it is clear that the stories generated with the two slightly different context rule sets performed the best, with the Revenge sub-category of Context performing almost midway between **acceptable** and **good**. As expected, the stories produced with the Random rule set fared the worst. Surprisingly enough, those with a rule set directed towards Character Attributes did almost as poorly.

For the fAIble tests we show three tables that show the average number of points given by the test subjects for each question. The higher the number, the more positive the result is. Note that the questions were shortened in the tables to make them fit into a single line. See the list in the section above for the exact words used in all surveys.

The results in Table 1 show that the fAIble I composite average was 0.88 (less than **acceptable)**.

The same procedure was done for fAIble II and the results are shown in Table 2. The composite average was 1.01 (almost exactly **acceptable)**.

The results of fAIble III follow in the same format and are shown in Table 3.

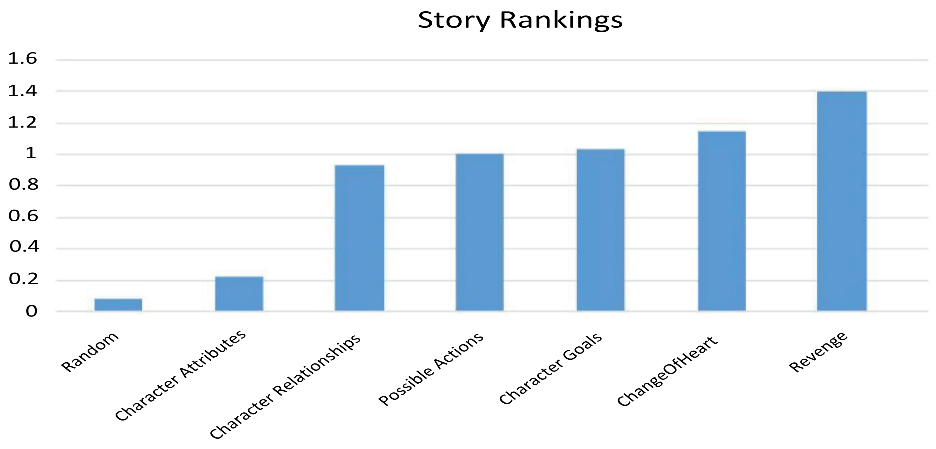


**Figure 9.** Results of the AESOP assessment.

**Table 1.** Tabulated results for fAIble I.

| Q# | Question Description | Mean |
|---|---|---|
| Q1 | Do story events appear to follow a coherent progression? | **1.39** |
| Q2 | Do characters appear to act based on internal reasoning and motivations? | **1.27** |
| Q3 | Do story events appear varied? | **0.98** |
| Q4 | Does the language resemble human generated narrative? | **0.29** |
| Q5 | Does the use of adverbs and adjectives add to the story's descriptiveness? | **0.88** |
| Q6 | Is the language varied across sentences and stories? | **0.46** |
|  | Average per Test Subject for all Questions | **0.88** |

**Table 2.** Tabulated results for fAIble II.

| Q# | Question Description | Mean |
|---|---|---|
| Q1 | Do story events appear to follow a coherent progression? | **1.00** |
| Q2 | Do characters appear to act based on internal reasoning and motivations? | **1.48** |
| Q3 | Do story events appear varied? | **1.12** |
| Q4 | Does the language resemble human generated narrative? | **0.79** |
| Q5 | Does the use of adverbs and adjectives add to the story's descriptiveness? | **0.93** |
| Q6 | Is the language varied across sentences and stories? | **0.75** |
|  | Average per Test Subject for all Questions | **1.01** |

**Table 3.** Tabulated results for fAIble III.

| Q# | Question Description | Mean |
|---|---|---|
| Q1 | Do story events appear to follow a coherent progression? | **1.11** |
| Q2 | Do characters appear to act based on internal reasoning and motivations? | **1.21** |
| Q3 | Do story events appear varied? | **1.00** |
| Q4 | Does the language resemble human generated narrative? | **0.66** |
| Q5 | Does the use of adverbs and adjectives add to the story's descriptiveness? | **0.93** |
| Q6 | Is the language varied across sentences and stories? | **0.57** |
|  | Average per Test Subject for all Questions | **0.91** |

Table 4 below compares the averages of the three fAIble systems. The description of the questions was further abbreviated to be able to fit all the data into one Table 4.

The average of averages for all fAIble versions is 0.933. As one would expect, there was improvement from fAIble I to fAIble II, although minimal for some of the questions. Unfortunately, the fAIble III results were between those of I and II. More on this below.

**Table 4.** Comparative results – Averages.

| Q# | Question Description | fAIble I | fAIble II | fAIble III |
|---|---|---|---|---|
| Q1 | Coherent progression? | **1.39** | **1.00** | **1.11** |
| Q2 | Reasoning/motivations? | **1.27** | **1.48** | **1.21** |
| Q3 | Variety? | **0.98** | **1.12** | **1.00** |
| Q4 | Human narrative? | **0.29** | **0.79** | **0.66** |
| Q5 | Adverbs & adjectives? | **0.88** | **0.93** | **0.93** |
| Q6 | Language varied? | **0.46** | **0.75** | **0.57** |
| Average/Test Subject – all Questions | | **0.88** | **1.01** | **0.91** |

## *Discussion of results of the independent tests*

The AESOP assessment focused on developing insights into what technique worked best. As expected, the results for the most complex set of rules (the rule set that used context – both *Change-of-Heart* and *Revenge*) performed the best. The surprising aspect of these results was the poor performance of the test stories generated with the Character Attributes module. This was particularly surprising in light of the fact that the stories generated with the Possible Actions module performed significantly better.

One possible explanation could be that the stories with the Possible Actions module, although ignoring character attributes, relationships and goals, at least had entirely plausible actions, even if there was no clear direction to the story. The readers were not told what the character's attributes, relationships or goals were, so to them, all actions seemed equally likely for all characters. In contrast, the module using character attributes attempted to bias the story towards events consistent only with character attributes. The hope was that the readers would infer what those attributes were that influenced which actions the character took. However, this may have been unsuccessful, given the very limited attribute and action sets that were used in the AESOP prototype tested.

The main insight that can be drawn from the above hypothesis is that a combination of attributes, relationships and goals is better than just character attributes alone. Character attributes in the presence of character goals allow characters to approach their goals realistically. However, character attributes without goals cause the characters to hardly do anything noteworthy because there is no incentive to do anything beyond what they're already good at doing, resulting in little character development and less diversity in the story.

In the fAIble assessments, the results show general improvement from fAIble I to fAIble II. However, the improvement is not reflected in all the questions asked, and both prototypes perform better than the other in at least one question. The performance of fAIble III was perplexing, however, as it seemed to regress. We cannot with any certainty explain this in light of our strong opinion that the stories generated by fAIble III were far better than those of earlier versions. One possible explanation could be that the students' evaluations for fAIble III were done near the end of their academic semester when they were more likely to be focusing on term projects due and upcoming final exams. This may have caused them to react impatiently to the stories and the requirement to complete a survey. As there is no way to prove that at this time, we designed and ran additional set of tests to clarify the situation.

Nevertheless, one takeaway from the independent tests, as indicated by questions #4 and #6 in all three prototypes, is that natural language generation seems to be what will require the most attention in any future research.

## Comparative evaluation of the four prototypes

The best way to provide clarity on the comparative performance of the prototypes was to design and undertake one final summative evaluation where a set of test subjects is given one story from each prototype version and asked to rank the four versions according to how each story addresses what is asked in each question, assigning 1st place to the one that addressed it best, and 4th place to the one that addressed it worst, with 2nd and 3rd to those in between. Keep in mind that unlike in the independent tests described previously, in this evaluation the lower the rank indicates the better performance (i.e., 1st place is better than 2nd place). The specific stories presented were those used as examples for each of the prototypes in their respective descriptions.

There were two specific objectives of this assessment:

(a) **Provide a direct, quantitative comparison of stories generated by the four prototypes**: The human test subjects compared stories from the four prototypes at the same time with respect to the same six questions used in prior individual tests described in the previous section. A story from AESOP was included to bring in AESOP into the same evaluation metric as the other three versions. We expected the results for AESOP to be the worst, so this would also serve as a baseline control experiment.
(b) **User Interface Assessment**: The test subjects were asked to rate the two user interfaces – the book graphic and the narrating avatar.

A total of 63 test subjects participated. However, not all subjects answered all the questions, so the total count is indicated for each question in their respective tables. Because of the anonymity of the process, we did not collect any demographic information. We only stipulated that the test subjects had to be 18 years of age or older, but that was self-reported. However, being an on-line assessment (partly because of the COVID-19 situation), it was not possible to confirm or enforce this rule.

The results are reported in the following section.

## Results of the comparative evaluation

Tables 5–10 show how the different versions were ranked by the test subjects in the context of each of the six common questions, as answered from the standpoint of adults. Note that each version is awarded 1 point for being ranked first; two for being ranked second; three for third and four points for fourth. As a reminder, the lowest Mean value indicates the version that performed best. The versions are presented in the tables in decreasing order of performance.

For Q1, as seen in Table 5, clearly the test subjects considered that fAIble III's story satisfied this statement best, followed by fAIble II, which performed narrowly better than fAIble I.

The test subjects also preferred fAIble III's story in the context of Q2, although not quite as strongly as for Q1. See, Table 6. Note that the fAIble I prototype outperformed the fAIble II version here, and is therefore placed above fAIble II in the table. This only happened in Table 6.

For Q3 (Table 7), once again, the story generated by fAIble III performed best, followed by fAIble II. Interestingly, the AESOP story performed better than fAIble I in the number of test subjects who selected it as #1 (although not in the mean ranking). AESOP actually performed better than we initially expected, with over 10% of the subjects ranking it first. Indeed, upon further analysis, the seemingly helter skelter actions of the characters of AESOP's story do seem to offer significant variability, which speaks to the core of this question.

For Q4 (Table 8), the test subjects overwhelmingly thought that the story generated by fAIble III best resembled human language. This is quite redeeming to our initial objective to improve the 'naturalness' of the fAIble III narrative text. It is interesting that one test subject thought the AESOP story to be the top one. This has to be considered an outlier.

**Table 5.** Q1: Story events appear to follow a coherent progression.

| Prototype Version | % Selected 1st Place | Number selected 1st | Mean | Std Dev | Count |
|---|---|---|---|---|---|
| fAIble III – The Saga of Martin | 50.9% | 30 | 1.83 | 0.98 | 59 |
| fAIble II – Q's Quest | 18.6% | 11 | 2.37 | 0.97 | 59 |
| fAIble I – Nathaniel's Revenge | 20.3% | 12 | 2.41 | 0.90 | 59 |
| AESOP – Harriette and the Witch | 10.2% | 6 | 3.39 | 1.01 | 59 |

**Table 6.** Q2: Do the characters appear to act based on some internal reasoning and motivations?.

| Prototype Version | % Selected 1st Place | Number selected 1st | Mean | Std Dev | Count |
|---|---|---|---|---|---|
| fAIble III – The Saga of Martin | 45.0% | 27 | 1.83 | 0.92 | 60 |
| fAIble I – Nathaniel's Revenge | 31.7% | 19 | 2.25 | 1.01 | 60 |
| fAIble II – Q's Quest | 15.0% | 9 | 2.37 | 0.84 | 60 |
| AESOP – Harriette and the Witch | 8.3% | 5 | 3.55 | 0.90 | 60 |

**Table 7.** Q3: Do story events appear varied?.

| Prototype Version | % Selected 1st Place | Number selected 1st | Mean | Std Dev | Count |
|---|---|---|---|---|---|
| fAIble III – The Saga of Martin | 49.1% | 28 | 1.82 | 0.99 | 57 |
| fAIble II – Q's Quest | 31.6% | 18 | 2.05 | 0.94 | 57 |
| fAIble I – Nathaniel's Revenge | 8.8% | 5 | 2.77 | 0.82 | 57 |
| AESOP – Harriette and the Witch | 10.5% | 6 | 3.35 | 1.00 | 57 |

**Table 8.** Q4: The language resembles a human-generated narrative.

| Prototype Version | % Selected 1st Place | Number selected 1st | Mean | Std Dev | Count |
|---|---|---|---|---|---|
| fAIble III – The Saga of Martin | 69.1% | 38 | 1.44 | 0.73 | 55 |
| fAIble II – Q's Quest | 16.4% | 9 | 2.11 | 0.75 | 55 |
| fAIble I – Nathaniel's Revenge | 12.7% | 7 | 2.65 | 0.77 | 55 |
| AESOP – Harriette and the Witch | 1.8% | 1 | 3.80 | 0.55 | 55 |

**Table 9.** Q5: The use of adverbs and adjectives add to the depth of descriptiveness.

| Prototype Version | % Selected 1st Place | Number selected 1st | Mean | Std Dev | Count |
|---|---|---|---|---|---|
| fAIble III – The Saga of Martin | 72.7% | 40 | 1.38 | 0.70 | 55 |
| fAIble II – Q's Quest | 12.7% | 7 | 2.25 | 0.72 | 55 |
| fAIble I – Nathaniel's Revenge | 14.6% | 8 | 2.51 | 0.81 | 55 |
| AESOP – Harriette and the Witch | 0.0% | 0 | 3.85 | 0.44 | 55 |

**Table 10.** Q6: The language is appropriately varied across sentences.

| Prototype Version | % Selected 1st Place | Number selected 1st | Mean | Std Dev | Count |
|---|---|---|---|---|---|
| fAIble III – The Saga of Martin | 71.2% | 42 | 1.41 | 0.74 | 59 |
| fAIble II – Q's Quest | 15.3% | 9 | 2.19 | 0.70 | 59 |
| fAIble I – Nathaniel's Revenge | 13.6% | 8 | 2.54 | 0.77 | 59 |
| AESOP – Harriette and the Witch | 0.0% | 0 | 3.86 | 0.47 | 59 |

The test subjects even more overwhelmingly thought that fAIble III generated the best narrative text in relation to the use of adjectives and adverbs for Q5 as indicated in Table 9. The performance of fAIble I and II was similar in that fAIble I had a larger number of first place votes (8) than did fAIble II (7). However, the latter had a lower mean ranking, and was therefore placed in second place in the table above.

Similarly, for Q6, as per Table 10, the test subjects overwhelmingly believed that fAIble III generated the most varied language. We note the same narrow margin between fAIble II and I.

In summary, for the first three questions dealing with plot progression, the story generated by fAIble III clearly was named best by most of the test subjects, with an average ranking of approximately 1.83. Similarly, for the last three questions that dealt with the quality of the narrative text (naturalness, variability, etc.), the test subjects also overwhelmingly gave the nod to the fAIble III story. These results are not ambiguous and resolve the ambiguities discussed earlier when comparing the independent test results.

## Discussion of results of the comparative summative evaluation

Table 11 depicts a summary of the comparative analysis of the four versions. It uses the average position over all adult test subjects for each question, and an overall average for all questions. Note that a low number indicates a higher ranking given by the test subjects, and thus reflects a better performance.

**Table 11.** Summary of comparative analysis.

| Prototype Version | Q1 | Q2 | Q3 | Q4 | Q5 | Q6 | Average |
|---|---|---|---|---|---|---|---|
| fAIble III – The Saga of Martin | 1.83 | 1.83 | 1.82 | 1.44 | 1.38 | 1.41 | **1.62** |
| fAIble II – Q's Quest | 2.37 | 2.37 | 2.05 | 2.11 | 2.25 | 2.19 | **2.22** |
| fAIble I – Nathaniel's Revenge | 2.41 | 2.25 | 2.77 | 2.65 | 2.51 | 2.54 | **2.52** |
| AESOP – Harriette and the Witch | 3.39 | 3.55 | 3.35 | 3.80 | 3.85 | 3.86 | **3.63** |

Clearly, the test subjects collectively strongly believed that fAIble III was the best of the four versions of our storytelling system, and AESOP was the least effective. The results also indicate an improvement every year of the project, as fAIble II was collectively ranked higher than fAIble I. These comparative results remove the ambiguities that arose from the individual testing of each prototype version indicated in Table 4 above.

Furthermore, there were many test subjects who offered extended (and well-considered) opinions on what they thought were the strengths and weaknesses of the system as a whole. A deep discussion about all of these is beyond the scope of this paper. Nonetheless, there were several weaknesses listed, and they generally pointed to the following flaws that will need to be addressed in future work:

(a) Repetitiveness. This complaint arose more often than any other. However, it is not clear whether the subjects meant repetitive wording or repetitive actions. Probably both.
(b) Stories did not always follow logically and sometimes made little sense. There was also insufficient justification for actions. fAIble III includes a mechanism for this justification to be done, but it did not come through well in the stories used in the survey.
(c) Stories are too violent for young children. This is true and we knew that going in; we made a comment to that effect in the Description of Research in the test subject surveys.
(d) Little introductory discussion or description of the geographical features are mentioned. The fAIble III backstory feature addresses this issue. However, while it provides a mechanism to define this backstory, insufficient knowledge seems to have been added to make this feature show through in a significant manner in the story used for these tests.
(e) Little character development. The mechanism for developing characters is there, but the stories did not seem to bring that out very well, possibly because of their short length.

Therefore, one important thing we learned is how to make the features present in the story generation system (e.g., backstory, plot hooks, action justification) better show through in the produced stories when testing in any future research.

In summary, we believe that our concept for an automated story generation system has great promise, not only as an entertainment medium but also as an educational tool to help children learn to read, as suggested by more than one test subject. We will explore this aspect in our future research.

## *Readability analysis of fAIble versions*

Several approaches for assessing the readability of a corpus of text exist in the literature. A full discussion of this topic can be an exhaustive undertaking in its own right, and thus beyond the scope of our research. Nevertheless, we believe that a strictly computational assessment for readability would be a good complement to the extensive human subject testing presented above. Therefore, we performed a limited set of such analyses to understand how our work measured up on an absolute scale and to prepare for when it is when compared with other competing narrative generation systems.

Many of the analytical methods employed in this analysis have decades of research backing them. However they are not intended as substitutes for human evaluations. The results collected in this section are meant to serve as additional insights on the effectiveness of the fAIble system. In order to ensure consistent readings, the stories that were computationally analysed with these methods were the same stories evaluated by the human test subjects in our human-based testing.

Readability tests are handcrafted formulas for measuring how ‘readable’ a text is. They leverage known and well-accepted statistical methods. These tests usually count words, syllables, characters, sentences, etc. These tests, however, are poor measures of the semantic complexity of a complete corpus, and make no efforts to rate its underlying structure. Despite this, they have seen widespread use in many text processing applications. Unfortunately, some of these applications oftentimes keep their exact implementations a trade secret. For this reason we used an open source library known as ***textstat***[2] for the Python programming language that implements many of the common readability tests.

We selected three tests to evaluate our stories. In our judgement, these best reflect the range of readability tests. They differ in what features they consider to be important for calculation, thereby providing some diversity to the overall analysis.

### *The Flesch Reading Ease test*

This test, with two variations, is considered the most popular one in the literature. These tests are designed to score a piece of text on the US Grade Level reading scale (Kincaid et al., 1975). The two variations of this test are the Flesch Reading Ease test, and the Flesch-Kincaid Grade Level test. They were used as recently as 2019 by a Florida legislation to rate the readability of insurance policies.[3] The Flesch Reading Ease is computed via the following formula:

$$206.835 - 1.015\left(\frac{\textit{total words}}{\textit{total sentences}}\right) - 84.6\left(\frac{\textit{total syllables}}{\textit{total words}}\right)$$

The Flesch Reading Ease test models a ‘numerical content’ of a corpus. For example, short sentences that contain words with few syllables will receive a higher score, while longer sentences that contain words with more syllables will receive a lower score. This dichotomy models what is often perceived as initial difficulty of a text. A higher score means a piece of writing is easier to read. We selected this measure to determine whether improvements to the system vocabulary and sentence length made a noticeable difference in its readability.

The Flesch Reading Ease test variation was chosen over Flesch-Kincaid Grade Level test because of its more complex nature. Flesch-Kincaid is very basic in its scoring, as it merely rates a story based on total words, sentences, and syllables. We did not use both to avoid recalculating a similar measure that served generally the same purpose. The scores in Flesch Reading Ease test range from 0–100. The table interpreting these scores can be found in Table 12:

**Table 12.** Flesch reading ease test rating index and interpretation (Kincaid et al., 1975).

| Score | Grade level |
|---|---|
| 0–10 | Professional |
| 10–30 | College Graduate |
| 30–50 | College Undergraduate |
| 50–60 | 10th – 12th grade |
| 60–70 | 8th & 9th grade |
| 70–80 | 7th grade |
| 80–90 | 6th grade |
| 90–100 | 5th grade |

## *Gunning Fog*

The Gunning Fog test is another commonly used test for rating a corpus. It has a unique way of determining the complexity of a corpus. While most tests use syllables to indicate difficulty, the Gunning Fog test is based on the concept that the usage and commonality of a word is what makes it more or less readable (Gunning, 1952). What defines a complex word differs according to implementation. The version found in ***textstat***[4] (and employed here) uses the original definition: complex words are those that contain more than three syllables. This test was selected because it captures an aspect of language that the Flesch Reading Ease test misses. The formula used to compute Gunning Fog index is:

$$0.4\left[\left(\frac{words}{sentences}\right)+100\left(\frac{complex\ words}{words}\right)\right].$$

**Table 13.** Gunning fog index output mapped to US grade levels (Gunning, 1952).

| Gunning Fog Index | US Grade Level |
|---|---|
| 1 | 1st grade |
| 2 | 2nd grade |
| 3 | 3rd grade |
| 4 | 4th grade |
| 5 | 5th grade |
| 6 | 6th grade |
| 7 | 7th grade |
| 8 | 8th grade |
| 9 | 9th grade |
| 10 | 10th grade |
| 11 | 11th grade |
| 12 | 12th grade |
| 13 | College Freshman |
| 14 | College Sophomore |
| 15 | College Junior |
| 16 | College Senior |
| 17 | College Graduate |

The output indicates how many years of US Education one would need to read and understand the text. The index is shown in Table 13. While no readability test is perfect, and words with many syllables still get caught up in the vague ‘complex words’ definitions, we chose the Gunning Fog test as a similar, yet different metric to compare with the Flesch Reading Ease test reported above.

## Automated Readability Index

The Automated Readability Index (ARI) is yet another readability test that outputs a US Grade level score. This readability test was chosen because it does not rely on syllables, but instead scores the difficulty of text by counting the number of characters per word, rather than syllables per words (Senter & Smith, 1967). We believe that including this metric would further diversify the readability tests selected. The formula for ARI is:

$$4.71\left(\frac{characters}{words}\right) + 0.5\left(\frac{words}{sentences}\right) - 21.43.$$

See, Table 14 for the mapping of ARI test results to US Grade Level.

**Table 14.** ARI scores mapped to grade level (Senter & Smith, 1967).

| ARI Score | US Grade Level |
|---|---|
| 1 | Kindergarten |
| 2 | 1st & 2nd grade |
| 3 | 3rd grade |
| 4 | 4th grade |
| 5 | 5th grade |
| 6 | 6th grade |
| 7 | 7th grade |
| 8 | 8th grade |
| 9 | 9th grade |
| 10 | 10th grade |
| 11 | 11th grade |
| 12 | 12th grade |
| 13 | College student |
| 14 | Professor or Grad student |

## Results

The results were collected by running the fAIble stories through the three selected methods obtained from the textstat Python library. Table 15 shows the results of each version of the fAIble system for the three tests executed.

**Table 15.** Summary results for the executed readability tests on all fAIble versions.

| | Flesch Reading Ease | | Gunning Fog | | ARI | |
|---|---|---|---|---|---|---|
| | Score | Grade Lvl | Score | Grade Lvl | Score | Grade Lvl |
| AESOP | 58.65 | 10th–12th | 2.54 | 2nd | 8.5 | 8th |
| fAIble I | 98.01 | 5th | 4.14 | 4th | 8.6 | 8th |
| fAIble II | 89.95 | 6th | 3.49 | 3rd | 0.8 | K |
| fAIble III | 90.77 | 5th | 3.10 | 3rd | 1.3 | 1st-2nd |

The fAIble system stayed reasonably consistent with its readability score for its target audience. fAIble I, II and III all aimed to improve (i.e., simplify) the readability of the text generated, so these results line up with the development goals set by the early cohorts. All three either are in or border the 5th grade reading level according to this test. The 5th grade level is considered to consist of children in the 9–10 year old age group, which is somewhat older than the target demographic for fAIble, but close. AESOP scored much less desirably, with results such that it would require 10th to 12th graders to understand the story.

The results from the Gunning Fog index also show an increasing trend in ease of readability. While AESOP scored the simplest (2nd grade) of the four versions, this could be explained by the simplicity of the sentences used in AESOP. After the high simplicity score for AESOP, the simplicity score goes down (less simple/more complex → higher grade level), with fAIble I scoring in the 4th grade reading level. However, the simplicity increases linearly for the next two versions fAIble II and III), taking it down to the 3rd grade reading level. The average age for 3rd graders in the US is between 7–8 years old, which falls within the target age range of fAIble. With the Gunning Fog index, the gap between versions is not as wide as it was in the Flesch Reading Ease. This could be because of the lack of emphasis that the former places on total number of sentences and syllables.

The drastic differences in the ARI scores seem to match up well with the design decisions of the fAIble II and III. Both versions aimed to make the content of the stories more complex, but also reduced repetitive language to make the output more succinct in the terms and phrases used. AESOP and fAIble I both had many characters in the text, but fewer sentences directly resulting from the relative brevity of the stories produced. fAIble II and III bumped up the character count slightly, but also increased the sentence count. This line of reasoning might help explain the drastic score differences. The age range for kindergarten is between 5–6 years old, which places the final iteration of fAIble just below our target age of 6–8 years of age for fAIble.

### *Conclusion*

As can be seen from Table 15, the mappings show widely disparate grade level associations with the stories produced by the different versions as calculated by the different methods. This reflects the different ways of calculating the scores by the different tests. The most important thing to take away from the above table is that fAIble III, the latest version of our system, was rated suitable for 5th, 3rd and 1st/2nd grades respectively by the three tests, putting it in the ballpark for our 6–8 year old target age group.

## Summary and conclusions

The objectives of the fAIble system from the beginning have been to a) produce interesting and well-written stories that children would enjoy; b) reduce authoring effort to the greatest extent possible; and c) produce a diverse sequence of stories that are not repetitive. However, as we embarked in the research, we found that creating fantasy worlds cannot be done without some non-trivial amount of development work to define the features of the characters (e.g., humans, human-like beings who are able to do some otherwise physically impossible feats, or homo-morphic animals) as well as the geographical locations available for the stories to unfold. Similarly, we came to the quick realisation that fantasy tales nearly always involve some type of quest by a protagonist that has five stages, as per Booker (2004). Therefore, we found it impractical to overcome one of the elements of Campfire that we believed to be a flaw. Yet, our work did succeed in creating stories that avoided the extensive use of templates that Campfire employed, instead combining a limited use of templates with a stochastic process that can vary the events that take place in the story. Moreover, this approach avoids the planning system approach taken by many authors. The natural language generation also creates reasonably good text. Even though the human-based results indicated that the natural language was what needed the most improvement, the fact is that the mechanisms to do that already exist in the fAIble system – it needs to be more extensively exploited than it was during the tests. However, it is true that such exploitation can only happen at the expense of more authorial effort on the part of the human, something we also wanted to address.

So, at the stage of our progress with fAIble, it is clear that additional authorial effort is essential to achieve our story quality and diversity goals. Nevertheless, we see an opening if one could use other stories to automatically populate the story world by using machine learning to extract characters, locations, adjectives and adverbs from these stories.

## Notes

1. http://www.bay12games.com/dwarves/
2. https://github.com/shivam5992/textstat
3. http://www.leg.state.fl.us/statutes/index.cfm?App_mode=Display_Statute&Search_String=&URL=0600-0699/0627/Sections/0627.4145.html
4. https://github.com/shivam5992/textstat

## Acknowledgments

The research described in this paper was funded by the US National Science Foundation under their International Research Experience for Students (IRES) program via grant #1458272. We would also like to recognize the following members of the research team: Lucas Pasqualin, Max Waldor, Lucas Gonzalez, Lauren Hastings, and Yasmine Moolenaar. Although not co-authors of this paper, this research could not have been successful without their extensive and valuable contributions.

## Disclosure statement

No potential conflict of interest was reported by the author(s).

## Funding

This work was supported by the NSF [IRES - 1458272].

## References

Alvarez, M. J., Amaya, R. E., Benko, K. A., Martin, J. T., Knauf, R., Jantke, K. P., & Gonzalez, A. J. (2019). Hello, narratives: Character development in automated narrative generation. *Proceedings of the 32nd Annual Florida Artificial Intelligence Research Society Conference*. Sarasota, FL. AAAI Press. May 2019.

Ansag, R. A., & Gonzalez, A. J. (2019). State-of-the-Art in Automated Story Generation Systems Research. *Journal of Experimental & Theoretical Artificial Intelligence*, *31*(1) , 15–40. https://doi.org/10.1080/0952813X.2021.1971777

Booker, C. (2004). *The seven basic plots: Why we tell stories*. Continuum.

Bottoni, B., Moolenaar, Y., Hevia, A., Anchor, T., Benko, K., Knauf, R., Jantke, K. P., Wu, A. S., & Gonzalez, A. J. (2020) Character depth and sentence diversification in automated narrative generation. *Proceedings of the 33rd Annual Florida Artificial Intelligence Research Society Conference*, North Miami Beach, FL. AAAI Press. May 2020.

Breault, V., Ouellet, S., & Davies, J. (2021). Let CONAN tell you a story: Procedural quest generation. *Entertainment Computing*, *38*. arXiv preprint arXiv:1808.06217.

Brézillon, P. (2004). representation of procedures and practices in contextual graphs. *The Knowledge Engineering Review*, *18*(2), 147–174. https://doi.org/10.1017/S0269888903000675

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

Ekman, P., & Friesen, W. (1978). *Facial action coding system: a technique for the measurement of facial movement*. Consulting Psychologists Press. Investigator's Guide 2 Parts

Gatt, A., & Reiter, E. (2009). SimpleNLG: A realisation engine for practical applications. *Proceedings of the 12th European workshop on natural language generation ENLG '09*. Association for Computational Linguistics.

Goldfarb-Tarrant, S., Feng, H., & Peng, N. (2019). Plan, write, and revise: An interactive system for open-domain story generation. *arXiv Preprint arXiv*.

Grinblat, J., & Bucklew, C. B. (2017). Subverting historical cause effect: Generation of mythic biographies in caves of qud. *Proceedings of the 12th International Conference on the Foundations of Digital Games*. Hyannis, Mass.

Gunning, R. (1952). *The Technique of Clear Writing*. McGraw-Hill.

Haslum, P. (2012). Narrative Planning: Compilations to classical planning. *Journal of Artificial Intelligence Research*, *44*, 383–395. https://doi.org/10.1613/jair.3602

Hollister, J. (2016). A Contextual Approach to Real Time, Interactive Narrative Generation. (PhD Dissertation). University of Central Florida

Hollister, J. R., & Gonzalez, A. J. (2018, March). The cooperating context method. Modelisation et utilisation du contexte. *iSTE OpenScience*, http://www.openscience.fr/La-methode-de-cooperation-de-contextes

Hollister, J. R., & Gonzalez, A. J. (2019). The CAMPFIRE storytelling system – automatic creation and modification of a narrative. *Journal of Experimental & Theoretical Artificial Intelligence*, *31*(1), 15–40. https://doi.org/10.1080/0952813X.2018.1517829

Kazakova, V. A., Hastings, L., Posadas, A., Gonzalez, L. C., Knauf, R., Jantke, K. P., & Gonzalez, A. J. (May, 2018). Let us tell you a fAIble: Content generation through graph-based cognition. *Proceedings of the 31st Annual Florida Artificial Intelligence Research Society Conference*. Melbourne, FL. AAAI Press.

Kincaid, J. P., Fishburne, R. P., Rogers, R. L., & Chissom, B. S. (1975). *Derivation of new readability formulas (automated readability index, fog count, and Flesch reading ease formula) for Navy enlisted personnel*. Research Branch Report 8–75. Chief of Naval Technical Training: Naval Air Station Memphis.

Lebowitz, M. (1985). Story-telling as planning and learning. *Poetics*, *14*(6), 483–502. https://doi.org/10.1016/0304-422X(85)90015-4

Meehan, J. (1977). TALE-SPIN: An interactive program that writes stories. *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pp. 91–98.

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. J. (1990). Introduction to WordNet: An on-line lexical database. *International Journal of Lexicography*, *3*(4), 235–244. https://doi.org/10.1093/ijl/3.4.235

Pickett, G., Fowler, A., & Khosmood, F. (2015). NPCAgency: conversational NPC generation. *Proceedings of the 10th International Conference on the Foundations of Digital Games* Pacific Grove, CA.

Porteous, J., Charles, F., & Cavazza, M. (2013). NetworkING: Using character relationships for interactive narrative generation. *AAMAS*, 595–602.

Porteous, J., & Lindsay, A. (2019). Protagonist vs. Antagonist PROVANT: Narrative generation as counter planning. *AAMAS*, 1069–1077.

Riedl, M. O., & Young, R. M. (2010). Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, *39*, 217–268. https://doi.org/10.1613/jair.2989

Senter, R. J., & Smith, E. A. (1967, November). "Automated Readability Index". Wright-Patterson air force base: Iii. AMRL-TR-6620. Retrieved March 18, 2012

Turner, S. R. (1991). A case-based model of creativity. *Annual Conference of the Cognitive Science Society*, vol 13, pp. 933–937.

Wade, J., Wong, J., Waldor, M., Pasqualin, L., Jantke, K., Knauf, R., & Gonzalez, A. J. (2017, May). A stochastic approach to character growth in automated narrative generation. *Proceedings of the 30th Annual Florida Artificial Intelligence Research Society Conference*. Marco Island, FL. AAAI Press.

Young, R. M. (2000) Creating interactive narrative structures: the potential for AI approaches. *AAAI Spring Symposium on Artificial Intelligence and Computer Games*.