Validating the plot for Interactive Narratives games

Carolina Veloso
Instituto Superior Técnico
Lisboa, Portugal
carolina.veloso@tecnico.ulisboa.pt

ABSTRACT

The authoring of an interactive dialog in video games is an overwhelming and complex task for game writers. Developing an Interactive Narrative that balances authorial intent and players' agency requires frequent in-depth testing, and the limited range of tools to assist authors in verifying their story can limit the creation of more complex narratives. Through reviews of the existing literature, we discuss the challenges of Interactive Story design and provide a model consisting of a set of metrics for testing interactive dialogs. Using this model, we developed a prototype for the Story Validator. This debugging tool allows game writers to experiment with different hypotheses and narrative properties in order to identify inconsistencies in the authored narrative and predict the output of different playthroughs, with visual representation support. Using the Story Validator, we conducted a series of user tests to investigate whether the tool adequately helps users identify problems in the game's story. The results showed that the tool enables content creators to easily test their stories, setting our model as an essential step towards automated authoring assistance for interactive narratives.

Keywords

interactive narrative, authoring tools, game writing, twine

INTRODUCTION

The presence of an immersive story or narrative in video games is often a central part of game design [1, 2] As an attempt to combine interactivity and storytelling Interactive Storytelling (IN) is a form of nonlinear narrative that gives the game an element of choice and, consequently, allows the player's actions to alter the course of the story [3, 4].

Although IN has enormous potential, enabling the creation of interactive systems that combine player interaction and dynamic plots, its main limitation lies in the challenges that authors face when writing this type of stories [5]. Developing an IN where players can feel immersed and engaged involves making the player's actions and choices have a powerful influence on the direction of the narrative, which makes it challenging for the author to guarantee a well-formed story.

Most traditional approaches rely on extensive and rigorous playtesting to obtain information on how the players experience the narrative and what might need to be improved. Playtesting provides insightful information that is not anticipated during development, helping authors ensure that both the game's narrative and the player's behaviour are well balanced [6, 7]. However, obtaining quality feedback for a detailed analysis can be challenging, expensive, and time-consuming [8].

Various works have offered different solutions to handle narrative conflicts caused by user behaviour in-game. However, hardly any works have focused on letting the author simulate and question their narrative during development. Instead, they opt for online AI approaches (such as drama managers [9]) that, during gameplay, provide ways to dynamically adapt the narrative and resolve conflicts created by unintended player's actions. These approaches create changes to the narrative that the author might not have intended, making them lose control over their story.

On the other hand, most authoring tools, such as Twine, Tinderbox, while facilitating the creation of IN, they lack the proper tools to identify possible continuity errors, to keep track of specific narrative properties and to envision the output of playthroughs prior to human playtesting. Instead, these tools rely heavily on the author's intuition to foresee these challenges or on an exhaustive number of later playtests.

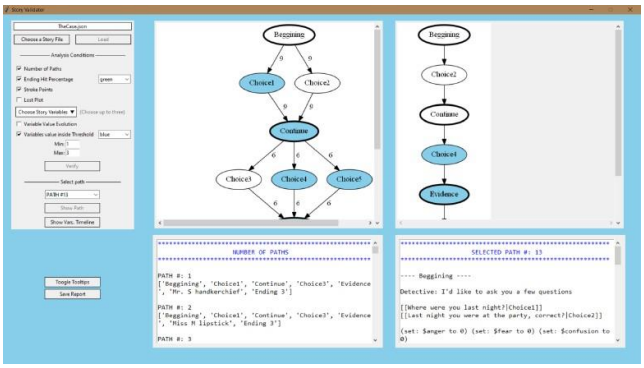


Figure 1: Story Validator tool interface

This work outlines the development of a tool that supports game writers in the creation of IN, whilst maintaining the human author's directorial control. Essentially, we strive to maintain the idea of authorial intent [10] and develop a system that allows human authors to

express their artistic intentions without feeling constrained. As a product of this work, we have designed, developed, and tested a prototype for the **Story Validator**. This tool takes a JavaScript Object Notation (JSON) file outputted from Twine and, by treating the branching narrative as a directed graph, uses a Depth-First Search (DFS) based algorithm to traverse all possible narrative paths and provide insightful data on different story metrics, and design issues encountered, via visual representations.

CONCEPTS

Before discussing related work and our solution to the problem, we will present the main concepts needed to understand this paper, in particular, definition of Interactive Narratives (IN) and common authoring challenges.

The Concept of Interactive Narratives

Interactive narratives (IN) are a form of nonlinear storytelling, where the player's actions and choices have a direct influence on the unfolding of the story. They function similarly to the Choose-Your-Own-Adventure storybooks [11], where the reader is faced with various decision points at which they must make a choice that alters the course of the story. At each decision point, the story branches in different directions, often leading to different outcomes.

Games with nonlinear narratives often have higher levels of interactivity, which facilitates the sense of causal-agency [12]. Causal-agency is the perception of being in control, meaning the player feels immersed in the virtual world as they believe that their actions have significant effects on the development of the narrative. In IN, instead of being tied up to one linear story, the player feels free to take their own path. This "freedom", however, is often an illusion. Each branch in the story has to be carefully tailored by the game's writer, who consequentially has to predict the different possible player behaviours that can affect the story at various states so that they can present those choices to the player. Nevertheless, narrative choice needs to be thought through carefully and authoring an IN is not done without a few challenges.

Authoring challenges in Interactive Narratives

The authoring of interactive stories in video games represent is an overwhelming and complex task for game writers, both in narrative design and implementation.

Next, we present a non-exhaustive list of challenges pertaining to the development of interactive narratives:

- **Authorial intent vs Player agency —** The most challenging problem with interactive narratives is the necessity to balance authorial intent and player agency in the context of storytelling [13]. Authorial intent is the trajectory that the game writer wishes the player follows, regardless of how the player acts during the game. Because interactive narratives allow for the user to interact freely within the story world, users often have the power to behave in ways that are inconsistent with the plot. This can either prevent the plot from advancing or make the player experience the story in a way that was not intended by the author.
- **Narrative exponential growth —** As the plot grows in complexity and the number of decision points increases, the authoring experience will often require substantial changes to keep the narrative coherent and can become overwhelming for the author [14]. Not to mention that the impact a choice has may end up only revealing itself in future states of the story, which is not always easy to predict.
- **Monitoring other game variables —** Besides what choices the player makes, the development of the story can be based on different variables and status, that are updated throughout to game. A common example are the karma systems, like in Fallout 3 [15], which consider the player's good and evil deeds, and shape the world around them as they progress. In more complex narratives, game variables can become difficult to keep track and their consequences may only unravel in later states in the game, which is not easy for the author to foresee during development.
- **Measuring the impact in user's experience —** The player playstyles are reflected not only in how they interact with the narrative but also on how they expect the story to unfold [16]. Due to the complex nature of interactive narratives, it is difficult for the author to predict the player's behaviour and their overall emotional experience. This is not only challenging to predict without prior human playtesting, as it is hard to ensure during development.

LITERATURE REVIEW

Interactive Drama

To maintain narrative coherency even when the player has freedom of choice, some past approaches have focused on the idea of integrating real-time AI methods to shape the narrative, allowing players to freely interact the game but, at the same time, making sure the narrative still follows a coherent structure. A popular example, first proposed by Bates [9], is the use of a **drama manager**. A Drama Manager (DM) is an omniscient agent that monitors the game world and guides the player's experience through the story, by searching for possible future plot points based on the evaluation of the current game world, while still allowing the players to interact freely.

For instance, one of the most famous examples of the implementation of a DM is the Mateas and Stern interactive drama Façade [17–19]. Façade uses its DM to monitor and update the simulation in real-time in response to text the user inputs by selecting the next story beat.

Figure 2: The interactive drama Façade

Like Façade, Riedl et al. presented the prototype for IN-TALE [20], where the agents are directed by the DM called **Automated Story Director**. The way the Automated Story Director handles user interactions is by maintaining a script of expected events and by planning out new narrative follow-ups to respond to the player's actions and achieve all concrete narrative goals pre-defined by the author.

The above-mentioned studies show that online AI approaches, particularly drama managers, are a popular solution to guarantee narrative coherence while allowing the player to freely interact with the game. Unfortunately, there has been hardly any work done regarding off-line approaches that attempt to identify possible narrative problems during development and allow the author to validate their narrative with different story metrics and predict the output of different playthroughs.

Interactive storytelling authoring tools

The authoring process remains one of the most significant challenges in the creation of interactive stories, and there is a need for authoring tools that both enable and assist authors in the creation of their content. Next, we present a study on different authoring tool for Interactive Narratives:

Tinderbox — Tinderbox[1], by Mark Bernstein [21] is a commercial spatial hypertext editor. The tool is mainly targeted at supporting authors' writing processes by creating and associating notes and ideas in the form of maps and charts. As they create their notes, authors can link and arrange them, building relationships between different text-based items.

Ultimately, while it provides many visual signifiers to utilize on individual nodes, this tool lacks metrics that inform the author on the possible outcomes and errors.

Ren'Py — The Ren'Py[2] engine, by Tom 'PyTom' Rothamel, is a software for creating visual novels. Ren'Py does not provide a visual representation of the story, such a node graph, and since a story has to be written directly on the text editor, if a project reaches a larger scale, the author might become overwhelmed by the amount of paths they have to monitor. Additionally, while Ren'Py includes a check script, called Lint, that runs through the project checking for errors, this system only includes errors in the project's python code and, according to the Ren'Py documentation page, "Lint is not a substitute for thorough testing" [22].

FAtiMA — Fearnot AffecTIve Mind Architecture (FAtiMA)[3] Toolkit, created by GAIPS, is a collection of tools and assets designed for the creation and use of cognitive and reactive agents with socio-emotional competences [23], and was developed to guide the emotional behaviour of the characters in the serious game FearNot! [24]. FAtiMA uses a character-based approach, meaning that the narrative is shaped by the way players interact with the agents and vice-versa. This differs from other popular game authoring tools, such as Twine, that are designed towards a plot-centric approach, which tends to restrict player involvement with the narrative to pre-defined key points [25].

Twine — Created by game designer Chris Klimas in 2009, Twine[4] is a tool designed to facilitate the creation of interactive stories with branching narratives. Twine's UI offers a bird-eye view storyboard, using a directed graph layout to create and visualize the narrative structure.

The creation of games with Twine, due to its simple design, requires only two elements: ***Passages*** and ***Links***. *Passage* is the Twine's terminology for the nodes on the story graph that players navigate through. Each *Passage* contains a block of text (known as lexia in hypertext theory) that is shown to the player when they reach that *Passage* during gameplay. *Passages* can also possess one or more **tags**, which function as labels that add information to the *Passage* but are not visible on the published version of the story.

Like branching narratives, where arcs connect nodes to each other, *Passages* are connected through *Links*, represented by an arrow on Twine's storyboard. To create a *Link*, the author needs to write the title of the *Passage* they want to link to surrounded by double square brackets (e.g. `[[Open the door]]`).

The author can also write multiple *Links* on separate lines inside a *Passage*, and because the player can only pick one at a time, this creates a branching point, meaning the story

splits into different paths. ***Path*** refers to the "route" the player has taken through the game's narrative. In Twine, the story branches at each decision point, so the choices the player makes throughout the game determine the path. At the end of the game, the narrative path includes all the *Passages* visited by the player during gameplay.

Besides which dialog options are chosen by the user, the path in which the narrative develops can also be dependent on different story's variables. These variables persist between *Passages* and store data that can be updated throughout the story. To test the values stored in a story variable at a certain point, Twine uses conditional statements (if, else-if and else), known as the <<if>> macros, which operate on a true or false logic.

Debugging and Troubleshooting

At the bottom of the Twine's storyboard page, there is a toolbar with both a "Test" and "Play" buttons, which open a playable version of the created story in a browser window, that the author can play. However, the "Test" button will also enable debugging facilities that help the author identify possible errors in the story.

It is important to note, however, that because of the way that Twine is designed, the author is obligated to play through the whole story in debugging mode to find every possible error or to consult all the metrics, since they can only view one Passage's state at a time. This means that testing a more complex interactive narrative, with multiple possible paths and endings, can prove to be a cumbersome and time-consuming task. Not to mention that the debugging mode lacks certain metrics that are essential for providing ways for the author to adjust their stories to their own narrative goals, such as: number of paths, number of endings, non-visitable Passages, endings' reachability imbalance, and others.

We have observed that many authoring tools lacked the proper tools that helped them analyse important narrative metrics and identify possible continuity errors.

Nonetheless, due to its wide adoption and easy to work architecture, we concluded that Twine was the most promising candidate for the incorporation of our model, which we will describe in the next section.

IMPLEMENTATION

In this section, we describe the design and development of the proposed tool. Figure 3 represents the conceptual model for our solution.

Authoring an interactive narrative

First, we chose to use Twine's authoring tool for the creation of the interactive narrative that will be used for testing. Because Twine is designed to develop and publish

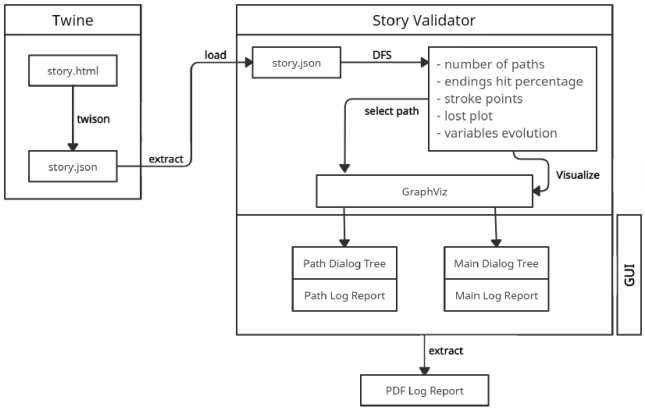


Figure 3: Conceptual model of our solution

interactive stories in the Web, all its data is encoded in a single HTML file. But while HTML is the default hypertext output for Twine, we found that exporting Twine's internal XML data in a JSON format would be an added value, for reasons of simplicity and easier processing. Therefore, we opted to use the story format Twison by lazerwalker [26] that converts the Twine's story data into a JSON document. This JSON file contains the data of each *Passage*, including the *Passages'* text, name, pid, *Links* and position on the Twine's storyboard.

After loading a JSON file into the Story Validator tool, if we parse it into a variable (e.g., story), we can then access the data inside. For example:

```
story.get("passages")[0].get("links")
```

gives us access to all the *Links* of the first *Passage*.

By treating the Twine's branching narrative as a directed graph, we can extract the information about the *Passages* and their *Links* from the JSON data and pass them as the tree's nodes and edges, and with that create a tree-like structure, similar to the branching dialogue tree graph that we built in Twine.

Functionality

In order to reach our goal, our solution needed to provide insightful data on different story metrics and identify design issues that could be encountered by players during gameplay. With this in mind, we opted to explore the tree graph using a **DFS algorithm**, since we wanted to explore the different *Passages* by following the same trajectory as a human player. While traversing through the tree graph, the algorithm gathers various metrics concerning the story, as follows:

- **Number of paths —** this metric enumerates all possible traversals of the story tree, including which Passages the player visits on each narrative path.
- **Endings Hit Percentage —** calculates the distribution of story's endings, to understand which ones are more likely to be reached.

- **Stroke Points —** we also needed to identify which plot points were common in all narrative paths. These refers to Passages, that regardless of the choices the player makes, are always reached.
- **Lost Plot —** this metric identifies narrative sections that, while plotted by the author, were never reached in an actual playthrough due to some design error. It should also notify the author of paths that end abruptly and do not each an ending.
- **Variables Evolution —** a branching narrative might contain different variables that the author needs to monitor. This metric keeps track of those variables and respective values throughout the different story paths.

In order to test their narrative, the author can use any of the previous metrics. Furthermore, it was important that our solution provided a way for the author to analyse any path in detail. Therefore, we made sure that the **Story Validator** would provide reports (according to the selected metrics) both of the overall story and of each chosen path, through visual representations (tree graph).

GUI application

Based on previous experience with the programming language, we decided to use Python to build a GUI using tkinter. Figure 4 presents the GUI conceptual representation that was created for the Story Validator of which we describe each of its components as follows:

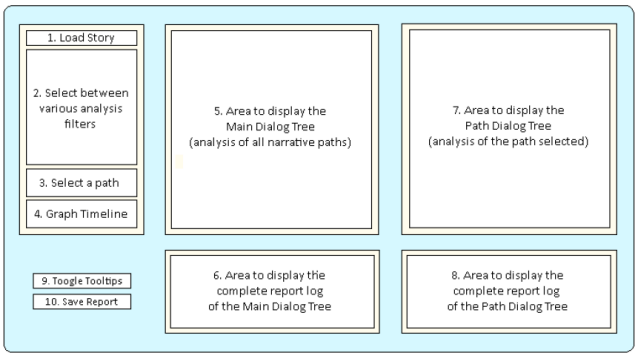


Figure 4: The Story Validator GUI conceptual structure

1. **Load Story —** Opens a file dialog window in tkinter that asks the user to select a story file (in JSON format) to be analysed by the tool.
2. **Analysis Conditions —** The user can select one or more options from a selection of analysis conditions. Each one of these conditions will influence different visual aspects of the Dialog Trees (5 and 7) and what information is shown on the Main Log Report (6).
3. **Select Path —** From a drop-down menu, the user can select which story path they want to analyse in detail. The path selected will appear on the Path Dialog Tree (7) area.
4. **Graph Timeline —** We use Python's Pyplot library to plot a dotted chart with the changes of the values of the story variables selected (throughout the path selected (3)), where the y-axis corresponds to the variable's values, and the x-axis corresponds to the Passages visited in the path selected.
5. **Main Dialog Tree —** We use GraphViz's DOT language to draw a dialog tree where each node represents a *Passage* in the story. If a Passage has Links to other Passages, they are represented in the graph as edges. This tree shows all possible narrative paths in the story selected and updates according to the Analysis Conditions (2) selected.
6. **Main Log Report —** Displays textual information regarding the overall narrative, according to the Analysis Conditions (2) selected.
7. **Path Dialog Tree —** We use GraphViz's DOT language to draw a second dialog tree where each node represents a *Passage* in the path selected (3). This tree also updates according to the Analysis Conditions (2) selected.
8. **Path Log Report** — Indicates what path was selected in (3) and prints the complete interactive story text into the log as well.
9. **Toggle Tooltips** — We use vegaseat's tooltip class to create a tkinter's tooltip widget that appears when the mouse is above the widget and explains what each Analysis Condition does for first time users. Then, we added the possibility for the user to switch these tooltips on-and-off by pressing the Toggle Tooltips button.
10. **Save Report** — We create and print a report that contains all the tests performed regarding the current visualisation, including what Analysis Conditions (2) were checked, the story variables and story path (3) selected, both dialog trees (5) and (7) and corresponding logs (6) and (8) and the Graph Timeline. This is accomplished by generating a PDF file using the PyFPDF library for Python. These PDFs can be used for a clearer reading of the results and for comparing validations with different conditions selected.

Using the tool to identify common IN problems

As stated previously, the main objective of this work is to support game writers by working as a debugging tool. Therefore, the tool performs a series of tests and reports on possible narrative problems:

Keeping track of Passages visited — The tool tells the user how many different paths the player can take, along with which *Passages* are visited on each path. The Main Dialog Tree (5) displays a directed tree graph with all the narrative paths that the player can possibly take, giving the author a general idea of how the story flows. To make a more in-depth assessment, the Main Report Log (6) displays which

Passages are visited in each path. Furthermore, the user can pick a path to analyse in detail (3), and the path will be displayed on the Path Dialog Tree (7), as a directed tree graph.

Moreover, the tool is also able to identify **Stroke Points**. We define Stroke Points as *Passages* that are visited in all possible narrative paths, meaning that regardless of the choices the player makes, they always end up reaching these Passages. This can be part of the designer goals, as it can be useful to ensure some parts of the story is always conveyed to the player. However, it may happen that the author does not want to withdraw the player's ability to choose, in which case Stroke Points become a problem. While travelling through all narrative paths, the tool's algorithm keeps track of the Passages visited. At the end, Passages that are visited in all narrative paths will be marked as Stroke Points. If the user has the analysis condition Stroke Points selected, the Main Report Log (6) will print out which Passages are visited in all paths. Moreover, on the Dialog Trees (5 and 7), Stroke Point nodes will have a bolded outline for easier identification.

Keep track of endings' reachability — Depending on the choices in dialog made, the player is led to different endings. However, it is difficult for the author to predict and monitor the distribution of those endings, without it being a laborious and time-consuming task. As a design objective, the author may want to create certain restrictions on the distribution of the endings, such as having an ending that is more common to obtain (i.e., has more paths that reach it than other endings), or even an ending with only one possible path.

If the analysis condition Endings Hit Percentage is selected, the Main Report Log (6) then provides the author with percentages on the likelihood of reaching each of the Ending *Passages*, as well as how many paths can reach each ending. Besides, on the Main Dialog Tree (5) the user can observe the distribution of the paths that enter each of the Ending *Passages* if the analysis condition Number of Paths is selected.

Keep track of story's variables — The tool identifies and keeps track of all story variables defined by the author and their values throughout each path. The user can also specifically select which story variables they wish to analyse (up to three). By selecting the analysis condition Variables Value Evolution, the user can observe the value changes of each variable selected, as well as the numeric value of each variable when an ending is reached.

By selecting the analysis condition Variables value inside Threshold, the tool highlights the Passages where the selected variables have a value between the MIN and MAX values, both defined by the user. Additionally, the tool can draw a graph with the value evolution of each variable selected, so the user can better keep track of the variables throughout the story.

Avoiding Dead-Ends and losing plot — A **Dead-End** is a *Passage* in the story that, once reached, prevents the player from continuing to play. These are different from an ending *Passage* since the latter corresponds to the end of the story. It is crucial for the designer to identify these cases, as they abruptly stop the player from continuing playing, however, doing so is difficult, due to the combinatorial nature of the exploration of the story.

As mentioned previously, Twine 2 supports the addition of tags in Passages. By taking advantage of this system, we defined that the author must attach the tag **ENDING-POINT** to a Passage to denote that Passage as an ending. Therefore, while traversing through the story, if it reaches a Passage where it cannot go any further, and that Passage does not have an ENDING-POINT tag, then the tool knows it has reached a Dead-End.

Furthermore, it is also important to identify if there are sections in the story that are never visited, regardless of how many times the player plays through the game and what choices are made.

If the user selects the analysis condition Lost Plot, the Main Report Log (6) will print out the *Passages* that were never visited as well as display which paths were not able to reach an ending *Passage*.

Figure 5: Example of an ENDING-POINT tag in Twine

AVALUATION

The following section presents the methods that were used to perform a series of **user studies**. These controlled experiments were conducted with the intent of:

- Finding out if the tool adequately helps the users identify problems in the game story.
- Determining whether users can operate the tool with ease and identify usability issues.

For this purpose, we conducted two phases of user testing, with a total of 25 participants. The data collected to uncover the usability problems in these studies were a variety of qualitative and quantitative data.

During **Phase 1**, we examined how the users feel about the tool's design and if it is easy and intuitive to use. During **Phase 2**, users were asked to use the **Story Validator** to

identify problems in a branching story with various design issues and suggest possible solutions.

Phase 1

During Phase 1, we performed a user test with a first prototype of our solution, to assess the usability of the tool. This prototype was an earlier version of the final application that was presented in the Implementation section, designed to evaluate the concept, and collect feedback from users to improve the tool. It differed from the final model in the fact that it did not include toggleable tooltips, the option to save a pdf report of the results, the story variables' values graphical timeline and it only allowed to test one story variable at a time. In addition, except for the background, the prototype had a black and white wireframe.

Performance

To perform the test, the users received a story (created using Twine) with no design problems to get familiarized with the tool and were then asked to perform 10 tasks using the **Story Validator** first prototype.

1. What is the total number of paths?
2. In PATH #7 what is the ending value for the story variable $anger?
3. In how many paths does the story variable $anger reaches an ending value of 0?
4. In what paths and passages does the story variable $anger has values between -4 and and -2?
5. Which ending is reached more often?
6. How many paths lead to Ending 2?
7. Which passages are visited in all paths?
8. Do all paths reach an ending?
9. Is there a passage that is never visited?
10. What is the text in passage Choice 4?

While the participants completed the tasks, we observed their performance and took notes. During this phase we used the think-aloud method, meaning that we asked the test participants to use the system while continuously verbalizing their thoughts. This helped us gather possible properties and design changes that the users might want to see in the updated version of the tool.

After completing the tasks, the participants were asked to rate how easy or difficult it was to solve each task through a Likert-scale based questionnaire. Additionally, participants were given a demographic questionnaire to help us identify the profile of our sample population and were asked to score the usability of the tool using the System Usability Scale (SUS).

Results and Discussion

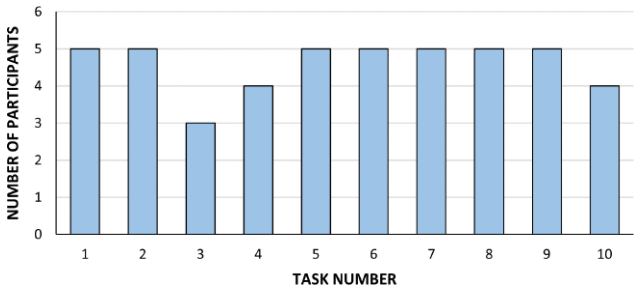


Figure 6: Task Completion Rate Results [Phase 1]

Total Completion Rates — Results show that two of the participants in the study group failed to complete Task 3. These incidences seemed to occur due to the rise in difficulty from tasks 1 and 2 to task 3.

Additionally, there was one participant that was not able to complete task 4, as they did not understand they needed to insert -4 on the MIN box and -2 on the MAX box, under the "Variable Value by given Threshold" section. Instead, they analysed one path at a time, searching for Passages where the variable $anger had values within that range. Eventually, they gave up, stating *"This is taking me too long. I feel like I am doing something wrong."*.

Finally, the same participant was unable to complete task 10, as they did not realise they could find what they were looking for under the Path Results Log. For all the previous tasks, the answers were under the Main Results Log, and because they were used to looking for answers to each task there, they did not realise they should look for the solution on the other log.

In conclusion, the completion rate for task 3 is **60%**, and for task 4 and 10 is **80%**. For all the other tasks (1, 2, 5, 6, 7, 8 and 9) the completion rate is **100%**, meaning every participant was able to complete them.

Task Level Usability score — After completing each task, the participants were asked to respond to a **Single Ease Question (SEQ)**, where they were asked how difficult or easy a task was to complete on a scale of 1 (very difficult) to 7 (very easy).

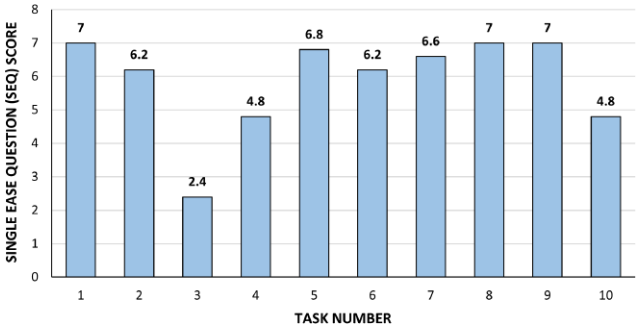


Figure 7: Average values of Single Ease Question (SEQ) [Phase 1]

The results show that the average user found Task 3 the most difficult, followed by Tasks 4 and 10. Tasks 1, 8 and 9 were the easiest to complete, with an average score of 7, according to all participants. These results match the ones in the "Total Completion Rates", as the tasks that some participants were unable to complete (3, 4 and 10) were the ones that were considered the most difficult.

Average Task Completion Time — During the experiments, we measure how long each user took to complete every single task.

The results presented show that most users took the highest time completing Task 3, spending an average of 104 seconds (almost 2 minutes) on it, followed by task 4 and 10, where users spent an average of 48 and 42 seconds, respectively. These results are equivalent to the ones found in the "Total Completion Rates" results and in the "Task Level Usability score" results. As expected, the tasks that were considered more challenging were the same ones that took longer to complete.

On the other hand, the task that took the least time on average to complete was Task 9, with an average of 3 seconds. After discussion, we concluded that this was because the response to Task 9 was in the same place as the response to Task 8, and therefore users took a short time to complete the task.

System Usability Scale — The average SUS score was **83.5** (SD = 9.45) which means that our solution, regarding its overall usability, is considered "passable". However, the participant P3 SUS score of **67.5**, suggests that the tool had some usability issues.

Usability suggestions — The following describes the different suggestions proposed by the participants regarding the tool's usability:

- **Help understanding how the tool works:** Some users reported having initial issues when trying to understand how the tool functions. While everyone was quick to learn, most still suggested with would had been easier had they been given a tutorial demo to explain the tool and how it worked. As one participant stated, *"Once you start clicking some check boxes you learn pretty quickly how* [the tool] *works [...] but having a help button or something similar would have helped a lot."* and added "[the tool*] is very overwhelming at first."*
- **Easier to read log report:** In cases where multiple analysis conditions were selected, users had issues navigating through the log report box, and often had to keep scrolling up and down to locate what they were seeking. Participants noted that if the box were bigger or if the analysis conditions were separate from each other, it would be easier to navigate.
- **Desire to analyse more than one variable:** One of the users (20%) reported the desire to view an analyse more than one variable at a time. While this did not hinder their ability to **complete** the tasks, their suggestion was noted as a hurdle that could arise in the future and therefore needed fixing.

Phase 2

During this phase, we performed a follow-up user test of the improved prototype, where participants were given a branching story (created using Twine) with various problems and were asked to use the **Story Validator** to identify those problems. They were then asked to propose solutions/changes to these problems.

Performance

The test story that was given for the participants to analyse had the following issues:

Problem 1: The Path #7 does not reach an Ending Passage.

Problem 2: The Path #8 does not reach an Ending Passage.

Problem 3: The Path #14 does not reach an Ending Passage.

Problem 4: The Passage "Ending 1" is never visited in any path.

Problem 5: The Passage "Choice 6" is never visited in any path.

All the previous problems can be identified in the Story Validator under the Analysis Condition "Lost Plot", which reveals *Passages* that cannot be visited and *Paths* that do not reach an ending.

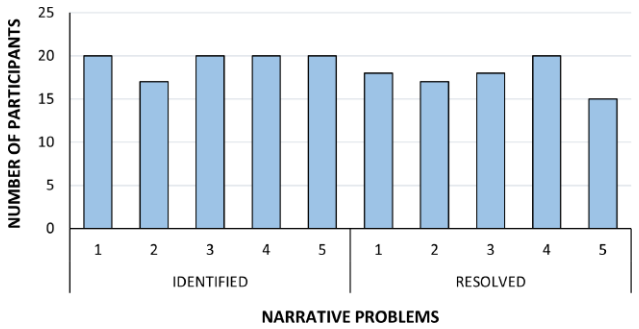
Throughout the study, we observed their performance, took notes, and measured the time each participant took to identify the problems, or until they gave up. Afterwards, the subjects were asked to score the usability of the tool using the SUS and to respond to a questionnaire.

Results

Total Completion Rate — We gathered that most participants were able to identify all the problems, except for three of them that were not able to identify Problem 2. We believe that the reason for the misidentifying of this problem was due to the fact that these participants in question, instead of using the Analysis Condition "Lost Plot" to identify the issues with the story, they found them by observing the Main Dialog tree directly. While this method is legitimate, none realized that while 4 paths reached the Passage "Mr S handkerchief" only 3 of them reached an ending. Consequently, none of these users solved the problem in

question. The completion rate for the identification and resolution of Problem 2 is **85%**.

Figure 8: Task Completion Rate Results [Phase 2]



Additionally, two users (**90%**) were unable to solve the Problems 1 and 3. Alternatively, Problem 5 seemed to be the one that was harder to solve, at least for five of the participants (**75%**). For all the other situations the completion rate was **100%**.

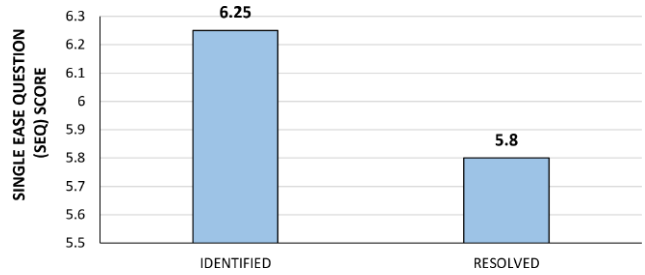


Figure 9: Average values of Single Ease Question (SEQ) [Phase 2]

Task Level Usability score — The average SEQ score for identifying the problems is **6.25** and for resolving the problems is **5.8**. While using the tool to pinpoint the problems was easy and only required a few "clicks", solving the problems required more energy to analyse the results and deduct a solution.

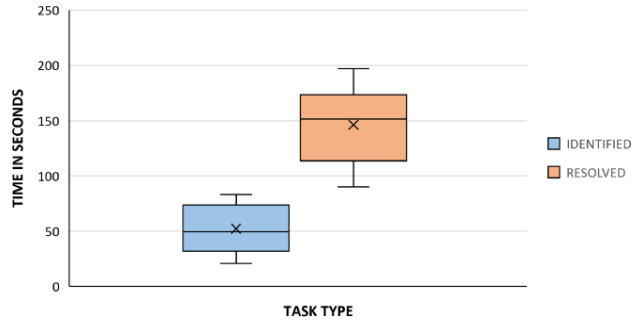


Figure 10: Task Completion Time for each task type [Phase 2]

Average Task Completion Time — Results show that when comparing the time it took users to identify all the problems, to the time it took solve them, the former took much longer. On average the time it took users to identify all the problems was **52.3** sec. (SD = 22.66), and to solve them it took them on average **146.2** sec. (SD = 31.84).

Overall, the time values for both finding and solving problems prove that the tool is efficient and that, with little habit, users can quickly use the tool to pinpoint and repair errors in their interactive narratives.

Usability Scale (SUS) System — On average the SUS score of our system is **92.4** (SD = 4.76). Our average SUS score proves that our system is considered a "truly superior product", however, we nonetheless received some suggestions for improvements on the tool.

Usability suggestions — The following describes the different suggestions proposed by the participants regarding the tool's usability:

- **Exploring using clickable Passages —** Some users suggested that, besides exploring the narrative through each path, it would be useful to have the option to explore using *Passages*. What they proposed was the possibility to interact directly with tree graph, by clicking on the nodes.

CONCLUSION

In this paper, we have underlined the current challenges concerning the authoring process of Interactive Narratives. After analysing several past works pertaining to the authoring of Interactive Narratives, we noticed how there was a lack of tools that provided ways for the author to test their narrative while considering the player's agency, during the game's developmental stage. More often than not, these works opt for online Artificial Intelligent (AI) approaches that, during gameplay, dynamically adapt the narrative and resolve conflicts created by unintended player's actions. This might lead to situations where the system takes control of the story, replacing human authorship.

With this work, we set ourselves to develop a tool for testing interactive dialogues for video games, that allows human authors to express their artistic intentions without feeling constrained. This approach has been designed to facilitate the development of interactive narratives in stages before human playtesting by letting the author explicitly test different hypotheses and narrative properties to identify possible design mistakes. The tool's GUI allows for a clearer picture of the interactive narrative authoring process, by providing a visual representation of the narrative structure through the use of directed graphs, that run through different test conditions.

After a thorough analysis of our test results, we concluded that we met our requirements. Several users said that they found the tool to be an essential asset for the creation of interactive stories, even though many expressed the desire to have more testing features and the option to interact directly with tree graph.

Overall, we believe that, as a first approach to this type of systems, our prototype managed to achieve the proposed objectives.

REFERENCES

1. E. Aarseth, "A narrative theory of games," in Proceedings of the international conference on the foundations of digital Games, 2012, pp. 129–133.
2. H. Qin, P.-L. Patrick Rau, and G. Salvendy, "Measuring player immersion in the computer game narrative," Intl. Journal of Human– Computer Interaction, vol. 25, no. 2, pp. 107–133, 2009.
3. M. O. Riedl and V. Bulitko, "Interactive narrative: An intelligent systems approach," Ai Magazine, vol. 34, no. 1, pp. 67–67, 2013.
4. S. Dinehart, "Dramatic play," 2009, online; Retrived 5-November- 2020. [Online]. Available: http://www.gamasutra.com/view/feature/4061/dramatic play.php
5. M. Mateas, "The authoring challenge in interactive storytelling," in Joint International Conference on Interactive Digital Storytelling. Springer, 2010
6. P. Mirza-Babaei, N. Moosajee, and B. Drenikow, "Playtesting for indie studios," in Proceedings of the 20th International Academic Mindtrek Conference, 2016, pp. 366–374.
7. P. Mirza-Babaei, V. Zammitto, J. Niesenhaus, M. Sangin, and L. Nacke, "Games user research: practice, methods, and applications," in CHI'13 Extended Abstracts on Human Factors in Computing Systems, 2013, pp. 3219–3222.
8. N. Moosajee and P. Mirza-Babaei, "Games user research (gur) for indie studios," in Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems, 2016, pp. 3159– 3165.
9. J. Bates, "Virtual reality, art, and entertainment," Presence: Tele- operators and Virtual Environments, vol. 1, no. 1, pp. 133–138, 1992.
10. M. O. Riedl, "Incorporating authorial intent into generative narrative systems." in AAAI Spring Symposium: Intelligent Narrative Technologies II, 2009, pp. 91–94.
11. B. Books, Choose Your Own Adventure Book Series. Bantam Books: NYC, 1979 - 1998.
12. C. Moser and X. Fang, "Narrative structure and player experience in role-playing games," International Journal of Human-Computer Interaction, vol. 31, no. 2, pp. 146–156, 2015.
13. R. Aylett, "Emergent narrative, social immersion and "storification"," in Proceedings of the 1st International Workshop on Narrative and Interactive Learning Environments, 2000, pp. 35–44.
14. E. Adams, Fundamentals of game design. Pearson Education, 2014, pp. 172–173.
15. Bethesda Game Studios, "Fallout 3," Rockville, Maryland: Bethesda Softworks, 2008.
16. S. Dow, B. MacIntyre, and M. Mateas, "Styles of play in immersive and interactive story: case studies from a gallery installation of ar fac¸ade," in Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology, 2008, pp. 373–380
17. M. Mateas and A. Stern, "Architecture, authorial idioms and early observations of the interactive drama fac¸ade," 2002.
18. [M. Mateas and A. Stern, "Fac¸ade: An experiment in building a fully- realized interactive drama," in Game developers conference, 2003, pp. 4–8.
19. M. Mateas and A. Stern, "Structuring content in the fac¸ade interactive drama architecture." in AIIDE, 2005, pp. 93–98.

20. M. O. Riedl and A. Stern, “Believable agents and intelligent story adaptation for interactive storytelling,” in International Conference on Technologies for Interactive Digital Storytelling and Entertainment. Springer, 2006, pp. 1–12.
21. M. Bernstein, “Collage, composites, construction,” in Proceedings of the fourteenth ACM conference on Hypertext and hypermedia, 2003, pp. 122–123.
22. T. Rothamel, “Quickstart - ren’py documentation,” 2020, online; Retrieved 20-December-2020. Available: https://www.renpy.org/doc/html/quickstart.html
23. S. Mascarenhas, M. Guimar˜aes, R. Prada, J. Dias, P. A. Santos, K. Star, B. Hirsh, E. Spice, and R. Kommeren, “A virtual agent toolkit for serious games developers,” in 2018 IEEE Conference on Computational Intelligence and Games (CIG). IEEE, 2018, pp. 1–7.
24. R. S. Aylett, S. Louchart, J. Dias, A. Paiva, and M. Vala, “Fearnot!– an experiment in emergent narrative,” in International Workshop on Intelligent Virtual Agents. Springer, 2005, pp. 305–316.
25. M. Cavazza, F. Charles, and S. J. Mead, “Character-based interactive storytelling,” IEEE Intelligent systems, vol. 17, no. 4, pp. 17–24, 2002.
26. lazerwalker/twison (2020). Retrieved 30 December 2020, from https://github.com/lazerwalker/twison