A Model for Stream-based Interactive Storytelling as a New Form of Massive Digital Entertainment

Marcelo Camanho, Bruno Feijó, Antonio L. Furtado
Dept. Informática, PUC-Rio
Rio de Janeiro, Brazil
{mcamanho, bfeijo, furtado}@inf.puc-rio.br

Cesar T. Pozzer
Dept. Eletrônica e Computação, UFSM
Santa Maria, Brazil
pozzer@inf.ufsm.br

Angelo E. M. Ciarlini
Dept. Informática Aplicada, UNIRIO
Rio de Janeiro, Brazil
angelo.ciarlini@uniriotec.br

***Abstract*— In this paper we present a highly scalable architecture for massive multi-user interactive storytelling systems based on video streams. The proposed architecture can support different demands for interactivity, generation, and visualization of stories in digital television environments, which include TV set-top boxes, tablets, smartphones, and computers. In this architecture, the same story adapts itself to the spectator's device in terms of rendering and interface processes automatically. Also a model for sharing massive interactive stories is presented. Moreover, the proposed system preserves the logical coherence of the story that unfolds while keeping it interactive.**

***Keywords*— Interactive Storytelling, Massive Digital Entertainment, Streaming, Cross-media games, TV.**

I. Introduction

Interactive storytelling as a form of massive digital entertainment has many open questions. How can we transform a TV program into a skilled storyteller capable of adapting the reactions of hundred thousand spectators simultaneously? How can we convert the audience into co-authors of a story in such a way that the evolving plot surprises even the original author of the script? These questions drive the attention of both TV producers and researchers. Since the 2000's, research works on interactive storytelling have been growing at a fast pace. However, the literature on interactive storytelling for digital television is still scarce, especially when we are looking for systems that can deal with a great variety of devices. Furthermore, most of the systems are based on a predetermined and fixed tree of possibilities. In the present paper, we propose a model and an architecture that address some of the above-mentioned issues.

In this paper we adapt the model used in our previous Logtell interactive storytelling system [1,2] to a new massive multi-user system. In the Logtell system, stories evolve in chapters and each chapter is a sequence of logically consistent events. Spectators interact with the story by making suggestions for the next chapter that will be accepted if coherence with the mathematical logic model of the story is preserved. Fig. 1 illustrates Logtell's evolutionary cycle of a story composed by two processing stages: plot generation and dramatization. A first draft of this story cycle was presented in a previous work by the authors [4] and a more complete version was recently implemented for the present paper. In the plot generation stage, a plan π and a list of suggestions for the next chapter are generated from the initial state s_0. The initial state is a set of rules and facts that define genre, characters, and general situations. A chapter is a plan π, which is a sequence of events to be dramatized. The user can pick up a suggestion from the list or write a new one. The execution of the events e_i produces a set of new facts to be added (*Facts*$^+$) and a set of facts to be excluded (*Facts*$^-$):

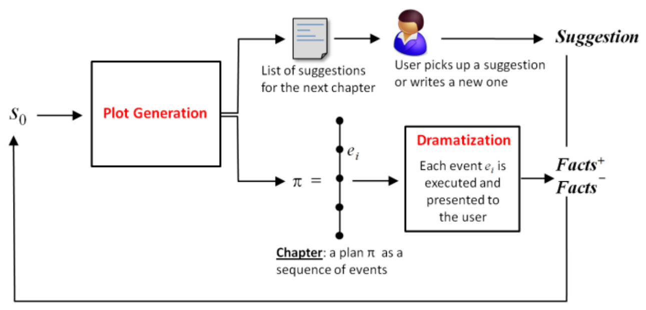

s_0
Plot Generation
List of suggestions for the next chapter
User picks up a suggestion or writes a new one
Suggestion
π =
e_i
Chapter: a plan π as a sequence of events
Dramatization
Each event e_i is executed and presented to the user
Facts$^+$
Facts$^-$

Fig. 1 The proposed story cycle

$$s0 = (s0 \cup Facts+) - Facts^{-} \qquad (1)$$

The list of suggestions is created by the same logical inference mechanism that generates the plan π. Therefore, the evolution of a story is always a surprise even for the original author. In the Logtell model, the inference mechanism is based upon a temporal modal logic framework [22] and a partial order planning algorithm. The inference mechanism is written in a constraint logic programming language [3] and details of a single-user system can be found in our previous works [1,2]. The prototype presented in this paper implements the same small sub-class of the popular Swords and Dragons genre used by those previous works, telling a simple story about damsels, heroes, villains, wizards, courage, revenge, and love. The logical rules of the story are also the same found in our previous works [1,2,4].

Our planning algorithm considers the following rules and facts that define the genre and the initial conditions of the story: (a) a set of goal-inference rules, which define situations that lead characters to pursue the achievements of goals (*e.g., if a damsel (i.e. a victim) is kidnapped, a hero will want to rescue her*); (b) a set of parameterized basic operators, with pre- and post-conditions (*e.g.* the operator *go(Character, Place)*, which indicates a character moving to a specific place, has simple preconditions, such as *Character is alive* and *Character is not kidnapped by someone*, and obvious post conditions, such as *Place is the new location of Character*); (c) logical descriptions of initial situations of the stories, introducing characters, their relationships, and their current properties (*e.g., Hoel is a hero*, *Brian is a hero*, *Marian is a princess*, *Draco is a villain*, and *Draco is a Dragon)*.

In our previous work [4], we propose a client-server model with **rich clients** (*i.e.* clients that provide rich functionality independent of the server). In that work, we assume that the spectators' devices could cope with heavy local processing. In the present paper, we take an opposite strategy; that is, we search for a flexible and scalable architecture for interactive storytelling in digital television that allows **thin clients** and a great diversity of platforms, as illustrated in Fig.2. Interactive storytelling systems for digital television should consider the fact that PCs, smartphones, digital TV set-top boxes, and tablets have different possibilities to render graphics and run artificial intelligence tasks. Moreover, those platforms have distinct screen resolutions, performances, and interfaces. In this paper, we propose a model of interactive storytelling for digital television that allows a better portability for different environments without losing the richness of interactivity and narrative coherence. We believe that our model and architecture can contribute for better solutions of broadcasting of interactive storytelling on digital television networks.

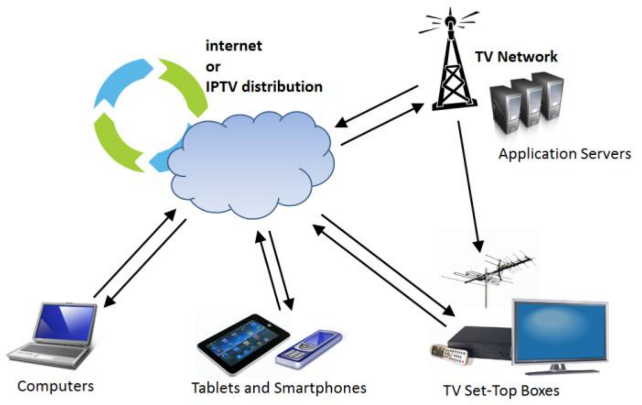


Fig. 2 Network environment of digital television for massive multiplayer interactive storytelling systems

The basic idea of our model is to transfer the plot generation and dramatization tasks of the chapter generation cycle (Fig. 1) from the clients to a cluster of servers. In this model, we have video streams as responses to the clients' requests. For this reason we classify our proposal as a Stream-based Interactive Storytelling architecture. The proposed model can cope with most of the strategies for spectators sharing interactive stories. The architecture based on this model adopts a light and highly responsive approach to web services, that is: HTTP as the transport layer, JSON [5] as the data layer, and REST [6] as architectural style. This approach is more flexible and lighter than the classic layers of XML as the data layer and SOAP as the protocol layer. Furthermore, the design of the architecture allows that different dramatization paradigms are easily adapted, such as text only, audio only (for visual impaired people), 3D rendering, video-based storytelling, 2D comics, 2D rendering, and even some combinations of them (*e.g.* text and 2D rendering). This approach to TV interactive storytelling allows a new form of massive digital entertainment, in which spectators can choose the type of dramatization they prefer.

This paper presents a contribution towards a distributed architecture for interactive storytelling that considers a multitude of environments and interfaces. As far as the authors are aware, there is no work in the literature that contemplates massive multiplayer interactive storytelling based on temporal modal logic and offers interactive dramatization on a variety of devices. Works on interactive storytelling usually focus on a single spectator experience or, even in the case of the multiuser systems, on individualized experiences similar to multiplayer games, which stray from the interactive storytelling proposal.

This paper is organized as follows. Section 2 presents related works. The architecture for a stream-based interactive storytelling system is presented in section 3. Processes and methods of this architecture can be found in section 4. Section 5 presents the interface of the prototype. A model of massive interaction is proposed in section 6. Conclusions are presented in section 7.

II. RELATED WORK

Independent, non-academic projects, like The Written World [7], show that there is a demand for multiplayer interactive storytelling as a product. This project, which is a web-based product financed by a crowd-sourcing agency, proposes an interactive storytelling system focused on an interactive narrative presented in a text format, where a user is "the world" and the other is "the hero". Another similar system is Sleep is Death [8], based on a two-player experience, where one is the author and the other the spectator. In this system, there is no backend to have an automated story generation – the author must prepare scenes in a graphical environment with interactive elements and the other user explores this scenario, similar to a "live" adventure game.

In Façade [9], the user plays a guest invited for drinks with a couple of friends. By using the mouse and by typing phrases, the user can interact with objects and talk to the characters using natural language, being able to influence the direction of the plot: the couple can fight, get together, or the user can be expelled from their house. Façade is regarded as one of the most successful interactive storytelling systems. However, it is based on a single player experience, and only available for computers.

In [10] experimentation was made to find out how multiplayer interactive storytelling systems could work. Similar to what happens with a single player interactive system, the problem of exception, that is, of something unexpected due to user interaction that was not predicted, was greatly increased. It uses a partially ordered planning algorithm to control the story generation and adaptation, with the addition of a "repairing" system to remove from the plan (the plot) elements that no longer makes sense after user interaction. This system differs from Logtell mainly because spectators play different roles, as players, while in Logtell the objective is to watch a story, being able to influence other characters, without the compromise of following and controlling only one of them.

One of the most similar projects to the model presented here is the Shapeshifting Media [11], which creates interactive screen-media narratives. It was broadcast on Finnish TV and provided an interactive experience that allowed spectators to participate by SMS messages, internet, and television. One of the main differences from our work is that it is centered on narrative objects, pre-recorded scenes, each defining their interaction points; that is: Shapeshifting Media system uses branching techniques, instead of a fully-fledged logical system, to create stories focused on their plot events definitions. In other words, it always uses fully instanced and pre-recorded events, instead of allowing more flexible plots to be constructed based on the logical structure of the story.

The approach presented by [12] uses StoryML, an XML based specification for multiplatform interactive stories, to build the TOONS application, featuring a story involving several devices: a TV screen, a light, an audio device, and a toy robot. It supports multiple environment presentation and interaction, what makes it similar to the architecture presented in the present paper. However, there are a couple of important differences from our system.

One of the key differences is that TOONS is based on presenting distributed media objects in multiple devices, where objects might even be something rather abstract like emotions. The approach of our paper is, however, to present a single story on multiple devices with different resolutions and interaction interfaces, through a continuous video stream. Also, TOONS has no real logical engine. Another difference is that the media objects must all be created beforehand, while in our system the plot can evolve from an initial state along unforeseen storylines.

Other works with multiuser interactive storytelling systems focus on local multi-player environment, as in [13], where users interact by drawing objects that are then transferred into the story in a virtual reality environment. In [14], multiple users inter-act with characters in different ways, using Wii videogame controllers, mobile phones, among others. However, those works differ from our work because they are focused on local interactivity instead of networked multiplayer systems.

There are already different storytelling researches that approach individual questions. On a general view, for an individual experience, Façade is the closest thing to the state of art in terms of interactive storytelling. However, it is based on a single player only experience, and only available for computers.

III. STREAM BASED INTERACTIVE STORYTELLING

In the proposed architecture, the main tasks of plot generation and dramatization of Fig. 1 occur at the server side. Instead of having the dramatization of stories being rendered and controlled on each platform, we propose video streams as responses to the clients' requests. Fig. 3 illustrates the proposed model, where the drama servers send video streams to the clients, which can be several devices in a variety of environments (e.g. windows computers, android smartphones, android tablets, etc.). This allows the same story to be easily generated in the same way for all the chosen environments; furthermore, the best image quality is guaranteed in all of them. Also, programming efforts will be much lower than the ones that would be necessary to have multiple versions of the software running in multiple platforms.

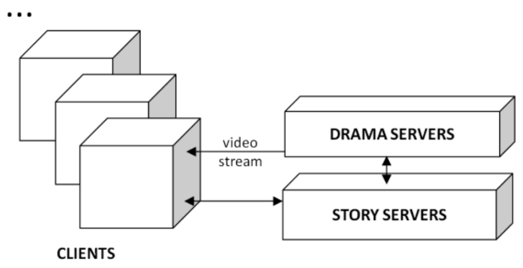


Fig. 3 Simplified stream-based interactive storytelling architecture

However, it is important to notice that some peculiarities apply to each environment. Also the ways of interacting with each one will not be the same. For instance, in a PC, users may use keyboard and mouse. On the other hand, in a smartphone, the screen and resolution are much smaller and, consequently, the interface cannot occupy the same size and be so full of details. As far as digital television systems are concerned, there are two distinct types: traditional television systems (terrestrial, satellite, and cable) and IPTV (Internet Protocol Television). Each of them requires a different client structure. For stories in multiuser mode, the same stream is shared through broadcast in a channel. In the case of Brazilian digital TV, for instance, the multiuser mode may be implemented through broadcasting together with a Ginga application, either in Ginga NCL [15], or Ginga-J [16]. The scenario is even more complicated, because

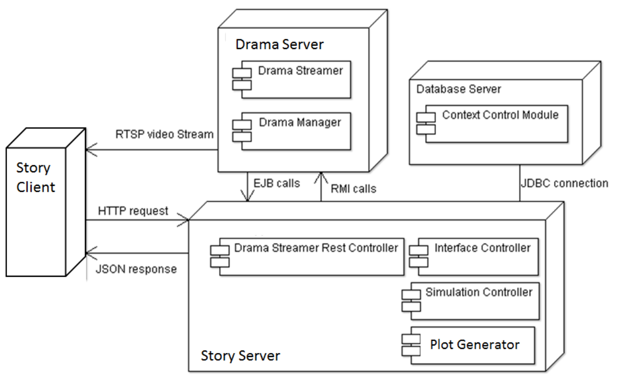


Fig. 4 The proposed architecture for the stream-based interactive storytelling system

smart TVs are using their own software environments.

Therefore, in our model, the choices to interact are defined in an agnostic way, and each client implementation is then responsible for interpreting it in the best way. Fig. 4 presents a more detailed architecture of the proposed model. Also this figure shows how the system is scalable. The Story Server environment, responsible for controlling story generation and adaptation, can be composed of multiple application servers, thus being able to support bigger loads.

The Story Client is capable of communicating with the Story Server through HTTP calls through a REST [6] interface. This is a much lighter approach than the use of EJB (Enterprise Java Beans) or Web Service calls, like it was done in our previous work [4]. This approach permits an easier creation of multiple client applications, since the HTTP protocol can be used for easy communication and transport of messages between the client and the story server.

The basis of the proposed architecture is defined by a module of the Drama Server called Drama Streamer. This module is responsible to intermediate the story that is being watched by different spectators.

The Drama Streamer module also follows a scalable strategy. There may be multiple instances in a given environment. The system uses the Story Server to find all the instances in a given cluster. Each drama streamer instance is capable of rendering one story at a time, since it captures the video of the server.

After some analysis with the possibilities of streaming with Java, we adopted the use of VLCj, a java version of VideoLAN – an open source video player, encoder, and transcoder. It is capable of supporting the media needs of an interactive story, and it was integrated into our system.

In the proposed architecture, the Story Client should connect to the Drama Streamer server using the RTSP protocol. The choice of VLCj for the client side was a natural one, since it was also chosen for the server side for encoding the videos.

For watching a story, a HTTP request is sent to a web server in order to discover which available Drama Streamers are free in the server cluster. This method call is made with a plain request, and a JSON [5] object is returned through the network. JSON objects are a standard data layer that simplifies interoperability between different layers of application, and tends to be simpler than other representations (such as XML).

The Plot Generator module is the core of the story creation: it is responsible for creating the list of events and suggestions that unfolds, using goal-inference rules and constraint logic, in a partially ordered planner algorithm. The Simulation Controller is responsible for generating the story itself and suggestions of strong interventions using the Plot Generator. Suggestions should be consistent and lead the plot towards different outcomes. This process is based on the story contexts that are stored in the Context Control Module, stored in the Database Server. These contexts define the story's logical rules and constraints, characters, relations, and the story's world initial state.

The Interface Controller controls interactions, organizing suggestions from the users. It coordinates the simultaneous dramatization and the presentation of suggestions for strong interventions in the various clients. It also controls the time during which users' choices are considered. The proposed architecture is completely different from our previous model with rich clients [4]. An interesting difference is that the control of interactions in our previous work [4] uses EJBs (Enterprise Java Beans) by the clients; now it is controlled by calls redirected from the REST interface.

IV. PROCESSES AND METHODS

In the prototype, we considered two platforms (windows and android) and two dramatization outputs (3D rendering using Unity [17] and 2D comics with text). When started, the drama streamer instance sets up its Story Output (by default the Unity3dOutput) and starts its VLCStreamer instance, responsible for streaming the story video for its clients. Afterwards, the Drama Streamer registers itself on its RMI Registry, which will be called by the JBOSS server of the Story Server environment. Then, the WEB calls that are received by the Web server will know to which server the RMI calls should go, when received in its REST interface.

After an available drama streamer is found, the client application should connect to this server. This connection actually occurs in more than one way. The dramatization of the story itself is received by a RTSP video stream, which is available on almost all modern platforms. Other protocols should be possible, but for now this one is being used since it is very portable and commonly supported.

Other than the video stream, Story Clients should also be coded to interact with the servers by using the REST interface. This interface contains all remote methods that need to be invoked in order to watch and control a story.

In Appendix A we present the supported methods in the REST interface of the Story Server (Drama Streamer Rest Controller). With only those methods and a video client that can connect the video stream, a new story client can be

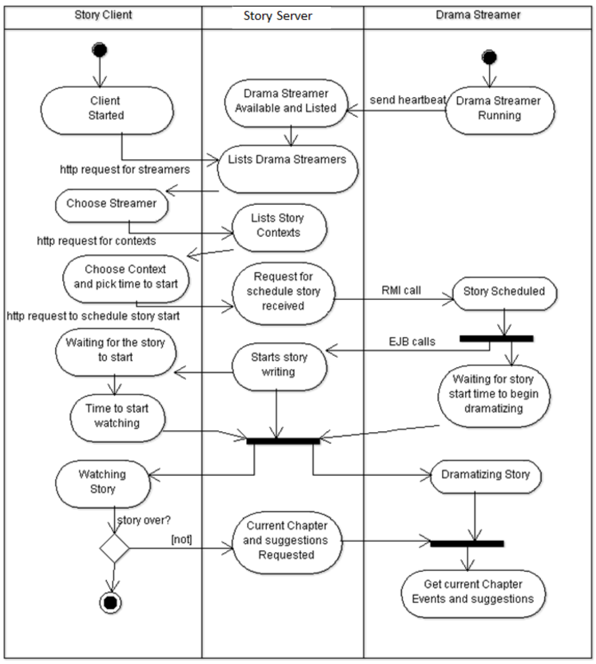

Story Client
Story Server
Drama Streamer
Client Started
Drama Streamer Available and Listed
send heartbeat
Drama Streamer Running
http request for streamers
Lists Drama Streamers
Choose Streamer
Lists Story Contexts
http request for contexts
Choose Context and pick time to start
Request for schedule story received
RMI call
Story Scheduled
http request to schedule story start
Waiting for the story to start
Starts story writing
EJB calls
Waiting for story start time to begin dramatizing
Time to start watching
Watching Story
Dramatizing Story
story over?
[not]
Current Chapter and suggestions Requested
Get current Chapter Events and suggestions

Fig. 5 Activity diagram for the process of watching a story

implemented in any platform.

Figure 5 is an activity diagram that shows how the activity of watching a story happens overall. We can see that the Drama Streamer starts and registers itself in the Story Server. Also we can see that all client calls go to the Story Server before going to the Drama Streamer. In this diagram we do not represent the video stream.

Basically, in this architecture, we can see that the client is just responsible for asking the user for preferences and then watching the story. All the story writing is still a responsibility of the Story Server. Any other remote method calls that the client needs are also received by the JBOSS web Server, which then redirects these calls to the Drama Streamer. Then, the Drama Streamer itself may need to make calls back to the JBOSS Application server, since it will ask, for instance, for the story writing process to start. The story writing process is still the same one used in our previous work [4], which can be accessed for further details.

In the proposed architecture there is a constant heartbeat being sent from the Drama Streamer to the JBOSS Server. This is a crucial aspect, because this is how the architecture will know which Drama Streamers are running, where, and what are their addresses. With this type of information, clients can know where to receive the RTSP stream of story, and the JBOSS server itself can do RMI calls to access specific Drama Streamer story context information and dramatization.

Client applications also connect to the drama streamer to reach the interaction options and chapters information. These calls are made by sending HTTP requests, receiving JSON object representations of the chapters, events, and available suggestions. By indirectly sending messages to the drama streamer, the interaction requests are also sent to the Story Server.

When we use the REST layer, the calls can be done by a simple HTTP request, instead of a more complicate access through Enterprise Java Beans calls. This approach facilitates the implementation and portability of the system to multiple platforms. In this case, the JERSEY library was used on the client side, although even if it did not exist, it was just a matter of doing HTTP GET method requests and deserializing the response in the form of JSON objects. This is a practice that can even be done natively on some languages, like Javascript.

In our prototype there is a small delay between the rendering, transcoding, and streaming of the stories. This limitation should not occur in an actual deployment environment in production use, with dedicated stations. Anyway, this should not be a big issue, since interactions in our system are mainly focused on the subsequent chapters – that is, while the user watches events of a given chapter, s/he is given the chance to suggest interactions for the next chapters. Also, in any TV broadcast process, it is inevitable that there is some delay, especially in the case of off-air television.

V. INTERFACE

Figure 6 shows two different start menus (on different operational systems and devices) representing the same interface objects and the same story controlled by a common server. These menus reflect different implementations of story

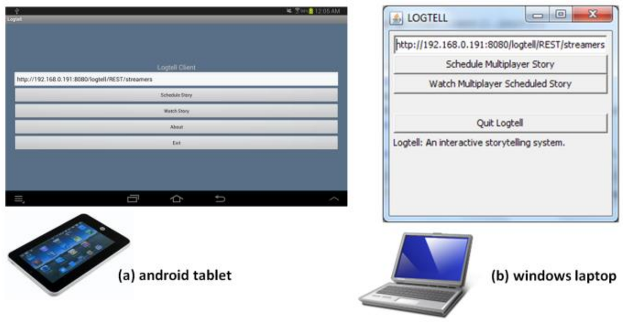

Logtell
Logtell Client
http://192.168.0.191:8080/logtell/REST/streamers
Schedule Story
Watch Story
About
Exit
LOGTELL
http://192.168.0.191:8080/logtell/REST/streamers
Schedule Multiplayer Story
Watch Multiplayer Scheduled Story
Quit Logtell
Logtell: An interactive storytelling system.
(a) android tablet
(b) windows laptop

Fig. 6 Two different story client menus (Android and Windows).

clients. One implementation is an Android application, running on a tablet, and the other is a windows application running on a laptop. Both applications are able to access the Story Server through the REST interface and to receive a video stream from the Drama Streamer Server. In the Windows application, VLC is used, but only as an embedded video player. In the android client, the native video player code was used, what shows how flexible the architecture is.

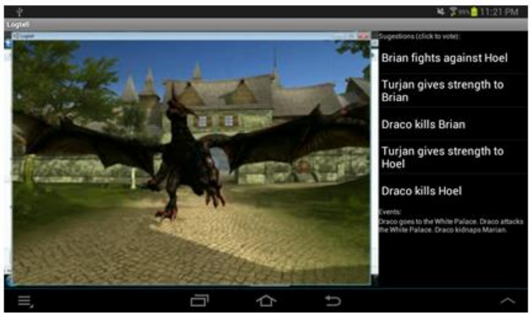
In Figure 7, we can see the same story running in two different applications, which are on different devices and operational systems. The difference between the two applications is the dramatization format, that is: one is a 3D rendering and the other one is a 2D cartoon with text. The last format has no animation. Several other formats can be easily implemented in the system, what is another evidence of how flexible the architecture is. Also there would be the possibility of having different forms to interact, such as using icons or voice input. As a design option, the proposed architecture cannot broadcast different dramatization formats simultaneously. Since these applications are running in different platforms, the interfaces can and should be different, because they have different screen resolutions and interaction devices. For instance, in the android application, the chapter description section is simplified.

VI. A MODEL FOR SHARING MASSIVE INTERACTIVE STORIES

The most challenging aspect of interactive stories on digital television is how to share desires and interventions from a massive audience on thin clients. We propose a model for sharing massive interactive stories that has two basic forms of user intervention: Local Weak Interaction and Global Strong Interaction.

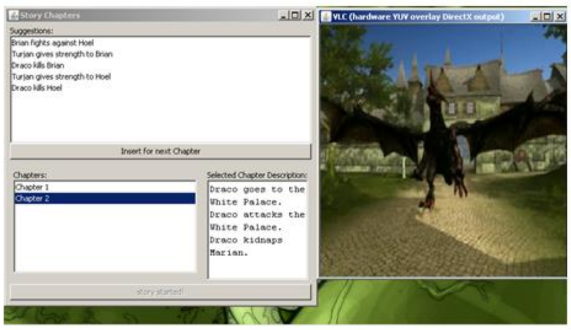
In the **Local Weak Interaction** form, the user interferes in the dramatization using local information and plug-ins. For example, the user may ask for another camera point-of-view. This form of intervention heavily depends on the processing capacity of the user’s device. Furthermore, this sort of intervention is not shared by other users and has no influence on the plot generation.

In the **Global Strong Interaction** form, the user may change the course of the story. In this form of intervention, we propose the following strategies: Most Voted Suggestion, Weighted Voting, and Harmonized Voting.



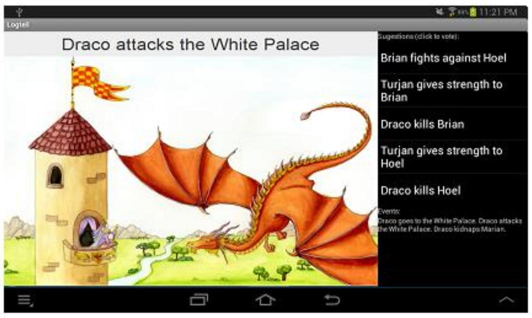
Sugestions (click to vote):
Brian fights against Hoel
Turjan gives strength to Brian
Draco kills Brian
Turjan gives strength to Hoel
Draco kills Hoel
Events:
Draco goes to the White Palace. Draco attacks the White Palace. Draco kidnaps Marian.

(a) android – Tablet – 3D



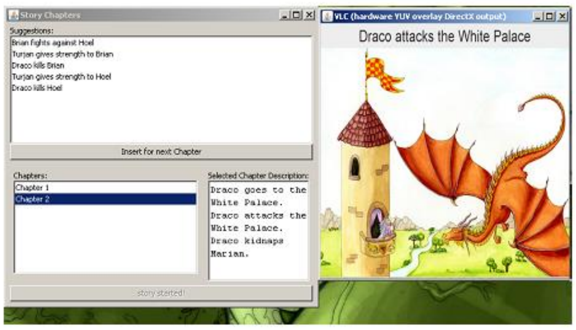
Suggestions:
Brian fights against Hoel
Turjan gives strength to Brian
Draco kills Brian
Turjan gives strength to Hoel
Draco kills Hoel
Chapters:
Chapter 1
Chapter 2
Selected Chapter Description:
Draco goes to the White Palace. Draco attacks the White Palace. Draco kidnaps Marian.

(b) windows – Laptop – 3D



Sugestions (click to vote):
Brian fights against Hoel
Turjan gives strength to Brian
Draco kills Brian
Turjan gives strength to Hoel
Draco kills Hoel
Events:
Draco goes to the White Palace. Draco attacks the White Palace. Draco kidnaps Marian.

(c) android – Tablet – 2D cartoon/Text


Story Chapters
VLC (hardware YUV overlay DirectX output)
Draco attacks the White Palace
Insert for next Chapter
story started!
Suggestions:
Brian fights against Hoel
Turjan gives strength to Brian
Draco kills Brian
Turjan gives strength to Hoel
Draco kills Hoel
Chapters:
Chapter 1
Chapter 2
Selected Chapter Description:
Draco goes to the White Palace. Draco attacks the White Palace. Draco kidnaps Marian.

(d) windows – Laptop – 2D cartoon/Text

In the **Most Voted Suggestion** strategy, when users ask the system to incorporate a specific suggestion, they are engaging themselves in a simple process of vote counting. The most voted suggestion will be chosen to be incorporated in the next chapter. The Interface Controller organizes the interaction with clients and interacts with the Simulation Controller as if there were a single user. In order to do that, it coordinates the simultaneous dramatization and the presentation of suggestions for global strong interventions in the various clients. Also it controls the time during which users' choices are considered. After computing the most voted suggestion, the Interface Controller checks whether the number of votes reaches a minimum threshold. If this is the case, the suggestion is sent to the Simulation Controller.

Weighted Voting is a strategy of global strong interaction in which the number of votes can be weighted by the potential of each option to trigger goal-inference rules. In this way, options that generate more interesting situations tend to be chosen.

In the **Harmonized Voting** strategy, compatible interventions can be combined in the same chapter. In particular, different groups of users may have different options. For instance, the groups can be separated by the characters they decide to support and, then, the planning algorithm (Plot Generator module) tries to combine the choices of all groups.

The way groups are formed to share a story is also an important issue. Off-air broadcast has no flexibility of choice, that is, users should join the story at specific date and time. In other systems of TV, such as IPTV, we have more options. The simplest strategy is to assume that any user is allowed to schedule the start of a story based on a specific context at a certain time. As soon as other users notice a specific story in a list of scheduled stories, they may be tempted to join the group. Users can either have equal rights to intervene in the story or not; in the latter case, different criteria can be established to assign their rights and privileges. Independent of the TV system (off-air, IPTV, internet TV), methods for the communication among users who share the same story can be adopted. In this case, users can discuss their interventions. An interesting model of incorporating social nets into an interactive storytelling system can be found elsewhere [18].

We consider the question of user interface modalities as an issue of utmost importance for massive interactive storytelling. Another work by the Logtell research group proposes a multimodal, multi-user and adaptive interaction model [19]. The experimental production of the configurable documentary "Golden Age" by the ShapeShifting TV research group [20,21] is a valuable source of information that could be implemented in the interface of our system.

During the tests of the prototype, we have considered the Most Voted Suggestion strategy only. Also, we implemented a simple interface modality in the prototype, as presented in the Figure 7. Moreover, usability issues from the viewpoint of

Human-Computer Interface were not considered in the present work. We made these decisions because the focus of the present paper is on the client-server architecture.

VII. EVALUATION

We made a partial performance evaluation of the proposed model in terms of delay time and number of users. Although the test is very simple, it represents a first measure of the capacity of the proposed multiuser system to handle loads without performance degradation.

TABLE I. PERFORMANCE TEST

With 1 Drama Server	**Number of Viewers**		
	1 User	***2 Users***	***3 Users***
Approximated Stream Delay	3.2 seconds	3.3 seconds	3.4 seconds

In the prototype, we used an i5 notebook running Windows 7 with 6 GB of RAM as the basic server machine. We implemented two separated servers running on the same machine, one working as a Story Server and the other one as the Drama Server. As clients, we used the following machines: (1) a 7-inch Android tablet running Ice Cream Sandwich with 1 GHz CPU and 1 GB RAM; (2) a Windows XP 1 GHz netbook; (3) a smartphone with 1 GHz CPU and 384 MB RAM.

As Table 1 shows, the prototype implementations do not change much in the delay of the video stream of the story. Also, this delay is mostly caused by the CPU intensive process of rendering the 3D story on the Drama Server, together with the live transcoding of the stream. On a real production system, better computers can be used. Moreover, even with a 3 second delay, it should be noted that this test reveals one of the most positive aspects of the proposed model: the same quality of story rendering would be much harder to achieve if all the story client implementations had to render the story by themselves.

Analyzing the test results, it seems fair to conclude that the model shows promising results towards the desired behavior of the multiuser system, where the Drama Server provides streams to multiple story clients and it is in charge of heavier CPU loads than those found in the clients.

VIII. CONCLUSION

In this paper, we present an extensible architecture that allows a multi-user stream-based interactive storytelling system to be available for different platforms. This architecture is currently already working and two different clients are available for the Windows and Android systems.

We show how a multiplatform stream-based storytelling experience can happen. By using video streams, we move the CPU cost from the clients to the servers, which allow the architecture to support more platforms, since processors can be quite slow on low end smartphones, tablets, and set-top boxes, for instance.

Another advantage of the proposed architecture is that it allows different types of Drama Streamers to be supported. That is, since the dramatization is taken from clients, it allows the same story to be rendered in different ways. In this particular aspect, a possible expansion of this architecture is to render the story in different graphical engines.

By making use of open patterns like REST and JSON through HTTP for the client server communication, the architecture presents itself as easily portable and extensible. The use of those patterns in APIs is an appropriate approach, because there are many libraries available to parse and process them. Also, in some cases, they are even part of native libraries of platforms. For instance, JSON is directly implemented in Javascript, where it origins from – thus making possible the creation of a new story client in pure HTML5, for instance.

For future works, different strategies of encoding will be investigated, for instance, generating multiple streams for the same story, in different formats and resolutions. With this alternative, the story clients could then have multiple choices that could be more appropriate for different platforms and screen sizes.

As research expands, more tests will be made regarding usability tests. Performance is also an important question, so for a real world use case, different optimization alternatives will be pursued, and tests with more powerful hardware should also be carried out.

Furthermore, it should be noted that since the architecture supports multiple platforms, the forms to interact can be expanded upon the different interfaces that they support. For instance, the mobile version could support voice recognition and motion sensing. This will be considered in our future works.

An important contribution of the present paper is the model for sharing massive interactive stories. As far as we are aware, no other work presents a model with strategies that are different from the simple “most voted suggestion”. The implementation and evaluation of all strategies we have proposed in the present paper are currently under development by our research group.

ACKNOWLEDGMENTS

This work was partially supported by CNPq (National Council for Scientific and Technological Development) and FINEP (Brazilian Innovation Agency), both linked to the Ministry of Science, Technology, and Innovation, CAPES (Coordination for the Improvement of Higher Education Personnel, linked to the Ministry of Education), and the Department of Informatics/PUC-Rio.

APPENDIX A

The supported methods in the REST interface of the Story Server (Drama Streamer Rest Controller) have special java annotations that define them as REST methods to be called over the HTTP interface. They are named accordingly, using the same name the java method has but separated by dashes. The same occurs to their parameters, showing how simple its access is. We decided to present the methods because they help the reader to understand the flexibility of the proposed system.

We describe bellow the available methods on the REST interface, for each URL format accepted:

- getDramatizationServers()
  http://[story-server:port]/REST/streamers
  Action: Returns the list of available Drama Streamer servers, their address and state (whether they are idle or already showing a story).

- getAllContexts()
  http://[story-server:port]/REST/streamers/get-all-contexts/
  Action: Returns the list of all available story contexts – used to choose which story to watch on the story server.

- scheduleContinuousStory()
  http://[story-server:port]/REST/streamers/schedule-continuous-story/{ip}/{selected}/{timetostart}
  Action: Schedules the selected context to be dramatized on the selected Drama Streamer interface. Since this is a multiuser system, the act of scheduling and joining a story to watch it are not necessarily single user exclusive.

- getTimeToStart()
  http://[story-server:port]/REST/streamers/get-time-to-start/{storyid}
  Action: Returns the time in milliseconds that a given story will still take to start, according to its schedule.

- getCurrentChapter()
  http://[story-server:port]/REST/streamers/get-current-chapter/{ip}
  Action: Returns the current chapter being shown on the Drama Streamer. This is important since the story client may need to get different data from the chapter and its events other than just the story visualization, which is provided by the video stream.

- getSuggestions()
  http://[story-server:port]/REST/streamers/get-suggestions/{ip}
  Action: Returns the available suggestions for the story (and chapter) being watched on the Drama Streamer. This is basically the main interaction mechanism in this version of story, where a user can vote for what he or she wants to happen in the story – this list is built by the system based on its logic [4]

- requestSuggestionInsert()
  http://[story-server:port]/REST/streamers/request-suggestion-insert/{ip}/{suggestionid}
  Action: Inserts a vote for the selected suggestion for the story being watched on the selected Drama Streamer. Since this is a multiuser system, more than one user can vote for different suggestions, where the most voted will be chosen and then be used for the following chapters of the story being watched.

REFERENCES

[1] CIARLINI, A. AND FURTADO, A. Understanding and Simulating Narratives in the Context of Information Systems. In Proc. ER'2002 – 21st. International Conference on Conceptual Modelling, Tampere, Finland, Oct. 2002.

[2] CIARLINI, A.E.M., POZZER, C.T., FURTADO, A.L., FEIJO, B.: A logic-based tool for interactive generation and dramatization of stories. In: Proceedings of the International Conference on Advances in Computer Entertainment Technology, Valencia, pp. 133-140 (2005)

[3] CARLSSON, M. AND P. MILDNER, P. SICStus Prolog - The first 25 years. Theory and Practice of Logic Programming, 12(1-2):35-66, 2012.

[4] CAMANHO, M.M., CIARLINI, A.E.M., FURTADO, A.L., POZZER, C.T., FEIJO, B., 2009. A model for interactive TV Storytelling. In: VIII Brazilian Symposium on Games and Digital Entertainment, Rio de Janeiro, Brazil (SBGames 2009), pp. 197-206, 2009. [DOI: http://doi.ieeecomputersociety.org/10.1109/SBGAMES.2009.31]

[5] CROCKFORD, D. Introducing JSON, 2006. Available at: http://www.json.org [Accessed 13 February 2013].

[6] FIELDING, R.T. AND TAYLOR, R.N., Principled design of the modern Web architecture. ACM Transactions on Internet Technology, vol. 2, no. 2, pp. 115–150, May 2002.

[7] KICKSTARTER. The Written World. Available at: : www.kickstarter.com/projects/thewrittenworld/the-written-world [Accessed 13 February 2013].

[8] ROHER, J. Sleep is Death (Geisterfahrer) – a storytelling game for two players. Available at: www.sleepisdeath.net [Accessed 13 February 2013].

[9] MATEAS, M., STERN, A., 2005. Structuring content in the Facade interactive drama architecture. In Proc. Artificial Intelligence and Interactive Digital Enter-tainment Conference (AIIDE 2005).

[10] RIEDL , Mark O., LI, B., AI H., and RAM A. Robust and Authorable Multiplayer Storytelling Experiences. Proceedings of the 7th Annual Conference on Artificial Intelligence and Interactive Digital Entertainment. Palo Alto, California, 2011

[11] URSU, MARIAN F., KEGEL, IAN C., WILLIAMS, DOUG, THOMAS, MAUREEN, MAYER, HARALD, ZSOMBORI, VILMOS, TUOMOLA, MIKA L., LARSSON, HENRIK and WYVER, JOHN. 2008. ShapeShifting TV: interactive screen media narratives. Multimedia Systems, 14(2), pp. 115-132. ISSN 0942-4962

[12] JUN HU AND LOE M. G. FEIJS. An adaptive architecture for presenting inter-active media onto distributed interfaces. In The 21st IASTED International Con-ference on Applied Informatics (AI 2003), pages 899–904, Innsbruck, Austria, 2003. ACTA Press.

[13] KUKA, D., ELIAS, O., MARTINS, R., LINDINGER, C., PRAMBÖCK, A., JALSOVEC, A., MARESCH, P., HÖRTNER, H. BRANDL, P., 2009. DEEP SPACE: High Resolution VR Platform for Multi-user Interactive Narratives. In: Proceedings of the 2nd Joint International Conference on Interactive Digital Sto-rytelling: Interactive Storytelling, pp. 185-196

[14] KURDYUKOVA, E. ANDRÉ, E. LEICHTENSTERN, K., 2009. Introducing Mul-tiple Interaction Devices to Interactive Storytelling: Experiences from Prac-tice. In: Proceedings of the 2nd International Conference on Interactive Digital Storytelling (ICIDS), pp. 134-139

[15] SOARES, L. F. G., RODRIGUES, R. F., MORENO, M. F., 2007. Ginga-NCL: the Declarative Environment of the Brazilian Digital TV System. Journal of the Brazilian Computer Society, Revista n. 4; v. 12; Mar. 2007 - ISSN 0104-6500.

[16] SOUZA, G. L. F., L. E. C. LEITE, BATISTA, C. E. C. F., 2007. Ginga-J: The Pro-cedural Middleware for the Brazilian Digital TV System. Journal of the Bra-zilian Computer Society. Revista n. 1; v. 13; Mar. 2007 - ISSN 0104-6500.

[17] UNITY Technologies. Unity. Available at http://unity3d.com [Accessed 13 February 2013].

[18] LIMA, E.S., FEIJO, B., POZZER, C.T., CIARLINI, A.E.M., BARBOSA, S., FURTADO, A.L., and SILVA, F.G., 2012. Social Interaction for Interactive Storytelling. In: Proc. of the 11th International Conference on Entertainment Computing (ICEC 2012), Bremen, Germany, pp. 1-15, September 2012.

[19] LIMA, E.S., FEIJO, B., FURTADO, A.L., CIARLINI, A.E.M., POZZER, C.T., SILVA, F.G., 2012. A Multi-User Natural Language Interface for Interactive Storytelling in TV and Cinema. In: XI Brazilian

Symposium on Computer Games and Digital Entertainment (SBGames 2012), Brasília, Brazil, November 2012.

[20] M. TSCHELIGI, M. OBRIST, and A. Lugmayr (Eds.): EuroITV 2008, LNCS 5066, pp. 40 – 50, Springer-Verlag, Berlin, Heidelberg, 2008.

ILLUMINATIONS, NM2: A Golden Age, ShapeShhifting Media Concept, 2009. Available at http://www.youtube.com/watch?v=jnMBrLWVwjE [Accessed 20 February 2013].

[22] CIARLINI, A.; VELOSO, P. ; FURTADO, A., 2000. A formal framework for modelling at the behavioural level. In: Proc. of the 10th European-Japanese Conference on Information Modelling and Knowledge Bases, Saariselkä, Finland.