A Genre-Independent Approach to Producing Interactive Screen Media Narratives

Marian F. Ursu[1], Jonathan J. Cook[1], Vilmos Zsombori[1], Ian Kegel[2]

[1] Narrative and Interactive Media, Department of Computing, Goldsmiths, University of London, SE14 6NW, UK
[2] Future Content Group, BT, Adastral Park, Ipswich, IP5 3RE, UK
m.ursu@gold.ac.uk

Abstract

In this paper we describe a production- and genre-independent approach to developing interactive and reconfigurable screen media narratives made with recorded essence. It is founded on a declarative language for the representation of interactive narratives called Narrative Structure Language (NSL). A complete software system for authoring and delivery and a number of productions realised with it, some in the public domain, validated the approach. This paper focuses on NSL and the authoring component of the software system.

Context

There are various approaches to interactive screen media narratives and they both influence and are influenced by the underlying computational models. Surveys of some of the most prevalent ones are presented in (Mateas and Sengers 1999), (Cavazza, Charles and Mead 2002), (Bulterman and Hardman 2005), (Riedl and Young 2006) and (Ursu et al. 2007a).

Two main categories of narrative systems may be identified with regards to the nature of the content: those that focus on virtual worlds and employ computer generated content and systems that work with pre-recorded material. In the first category, the audio-visual behaviour of the entities of the virtual worlds, such as the characters of the story, can be controlled, so they are naturally modelled as autonomous agents. Stories emerge from their interaction. The engagers – the active viewers – are given means to control some of the agents, either as external viewers or by becoming themselves characters in the story. Representative examples of narrative systems in this category include the work of Mateas, such as Façade (Mateas and Stern 2005) and the work of Cavazza, such as the system described in (Cavazza and Charles 2005).

Agents encompass goals and primitive actions. Their behaviour emerges as a sequence of primitive actions. Planning is used to drive this behaviour: a plan is a sequence of actions that could lead to the achievement of a number of goals. Planning could be used in a similar fashion – referring to primitive actions and goals – in the absence of encapsulating behaviour in autonomous agents, as it is illustrated by Mimesis (Young 2001). However, the resulting narrative systems display the same characteristics as the agent-based ones. They focus on modelling character behaviour.

The narrative systems in this category are more akin to games and the focus seems to be on micro-narratives – shorter episodes of interaction in a virtual story world. They do not display the narrative complexity say of a dramatic film production. Narrative mediation has been employed in most of this work in order to enforce some degree of narrative coherence (Rield and Young 2006), but the approach is inherently focused on a high degree of emergence. Consequently, the authoring style is distant from that employed in movie or TV programme making. Authoring refers to modelling the characters and other animated entities of the environment; the narrative is a consequence of the interaction. This approach may not be that appropriate for situations when there is a storyteller who has a story to tell. We focus on such situations.

In our work we concentrate on interactive productions made with pre-recorded material, such as video and audio clips, in established genres such as drama, documentaries and news (Ursu et al. 2007b). We aim for a quality similar to that of the non-interactive (linear) traditional programmes, including coherence, meaningfulness, information and aesthetic quality, and level of engagement and enjoyment. We take as starting point the workflow processes employed in traditional linear programming, by script writers, directors and editors, and extend them towards interactivity and non-linearity. At this end we devised a formal declarative language – the *Narrative Structure Language (NSL)* – and an associated *Authoring Environment*, in which interactive narratives could be conceived, sketched, realised and tested (reference concealed for blind review). They support and inform the thinking and mechanics employed in movie and programme making by directors and editors. The specifications made in NSL, whilst leading to interaction and reconfigurability, focus on the way narratives are edited from atomic elements, such as video clips, which we call *media items*.

Authors, in our approach, craft the narratives directly, by specifying their structures. Nevertheless, the productions themselves, being interactive, can lead to numerous narrations by adapting both plot and discourse to the choices of the active viewers or engagers. Different narrations are realised automatically at delivery time through the compilations of a *Realisation Engine* which consists, essentially, of an *Inference Engine* for NSL and a *Media Composer* (Ursu et al. 2007a). The individual narrations are compiled on the basis of an artefact – a specification made in NSL. We denote such an artefact as a *narrative space* and individual narrations as *narrative threads*. The narrative space is authored. Narrative threads are computed from it automatically by the Realisation Engine on the basis of the engagers' interactions. Each narrative thread is a number of sequences of media items (layers) that are played in parallel.

There is less emergence and more editing guidance in our approach than in the ones previously discussed, and the specifications made in NSL refer to both plot and discourse development. The computational models and the associated tools we developed employ different aspects of artificial intelligence: the representation language NSL has evolved through a process of knowledge elicitation; in turn, NSL is a language which can capture authoring expertise; various AI techniques have been employed in the implementation of NSL's Inference Engine; rules of thumb are being implemented in *semantic testing* algorithms, which are capable of identifying potential pitfalls of the authored narratives; last but not least, heuristics are also being implemented in a component of the Realisation Engine called the *Optimiser*, which is responsible for dealing with scalability to large productions, possibly large number of engagers and possibly less powerful delivery devices.

Computational models and systems have been developed for interactive narratives made with pre-recorded material. The interactive TV productions fall in this category. Essentially, they are linear, non-interactive narratives that can be viewed in extended time intervals, can be navigated forwards and backwards and can be chosen from a number of streams delivered in parallel. This may be called pseudo-interactivity or brute-force interactivity (Ursu et al 2007), as the productions themselves are not interactive, but some degree of interactivity is achieved through fast delivery channels and/or powerful storage and delivery devices. Our work is not in this category, as we target truly interactive productions. They are addressed in the research arena.

Terminal Time (Mateas, Vanouse and Domike 2000) and Mindful Camera (Barry 2005) have at their centre commonsense reasoning. Terminal Time is a narrative system that produces historical documentaries which can respond to the audience's appreciation whilst it is being delivered. It focuses on the automatic generation of the narrative text, through a combination of knowledge based reasoning, planning, and natural language generation. Video clips are subsequently attached to the narration through term matching based on TF/IDF. Terminal Time is essentially one production, within a genre whose narration is carried out mainly by the text. The narrative intelligence, save the CYC ontology, is hard-coded in bespoke software components implemented in LISP. Mindful Camera makes suggestions regarding what to shoot next to documentary film makers, so it does not address the issue of delivering interactive narratives.

The work of Gordon et al. (2004), Wages et al. (2004), and Agmanolis and Bove (2003) are examples of works that are the closest related to ours. They all employ the branching narrative paradigm that contains a fixed number of points at which the decisions or actions of the engager determine the way the narrative unfold. They are represented as directed graphs, in which nodes are scripted scenes and arcs correspond to possible transitions. In Gordon's system, which is a specific application, rich interaction could happen within nodes, but the nodes contain more or less independent episodes. Though in virtual reality, Wages proposes a production independent environment that supports the authoring of interactive narratives as directed graphs, in which nodes, in turn, could be directed graphs. However, there are no other significant authoring means apart from the recursivity of the specification. Finally, Viper (Agmanolis and Bove 2003) is a system that allows the creation of programmes that could change during viewing in response to preference or profile information, presentation devices or real-time sensor feedback. Automatic editing is carried out on the basis of editing guidelines, expressed via a number of primitives implemented in the programming language Isis, and media clips annotations, made via a dedicated interface. Viper is close in aim to our work, but, among other limitations, it provides neither a language nor an authoring environment for expressing narrative structures.

Two major results of our work make original contributions to the world of interactive storytelling: a genre independent representation language for interactive narratives (NSL), together with an associated inference engine, and a corresponding authoring environment. They are essential components of a complete system for authoring and delivering interactive and reconfigurable screen media productions. The narrative system has been validated through a number of productions, of which notable is Accidental Lovers (Tuomola, Saarinen and Nurminen 2006/2007), an interactive dramatic production broadcast on 12 occasions on Channel 1 of Finland's National Broadcasting Company. The remainder of this paper presents an overview of NSL and the authoring environment.

Narrative Structure Language (NSL)

Generalities. Recall that in our approach screen media narratives, such as movies and TV programmes, are automatically edited at "viewing" time from atomic media items, such as video and audio clips, to reflect the interaction received from the viewer(s)/engager(s) *during*

the delivery; each engager can receive a personalised version of the production, called a *narrative thread*, which corresponds to their preferences at the time of viewing. The automatic editing of narrative threads for each production is achieved on the basis of its *narrative space*, which is a specification made in NSL; each production has *one* authored narrative space.

Narrative threads are made of media items – a narrative thread is a number of sequences or juxtapositions of media items played in parallel; each sequence is a track or layer. For each delivery/viewing, the Inference Engine computes playlists in between consecutive interactions; each new interaction causes the Inference Engine to compute a new playlist. Playlists are passed onto the Composer, which accesses the essence and creates the actual media object which will be the content received by the engager.

Any (correct) narrative space is an NSL *narrative object*. A narrative object is a composition of other narrative objects. The *primitive* or *atomic* narrative objects are representations/descriptions of the pre-recorded media itself (such as video and audio clips); for simplicity, they may be regarded here as being the same with the media items (mentioned before). In order to support the sketching and design of narratives in the absence of high quality material, NSL also supports media items with no reference to essence, called *placeholders*. Media items are the basic building blocks of any narrative object. Narrative objects are composed into (more complex) narrative objects via three main NSL composition structures: *link*, *layer* and *selection group*. NSL imposes no restrictions with regards to the depth or order of the composition.

The remainder of this section gives on overview of NSL. The semantics of each composition structure is specified with regards to its interpretation into a playlist. Since narrative objects are recursive definitions, interpretation is inherently a recursive process. The interpretation process is carried out, for each interaction, until the compiled playlist consists only of atomic narrative objects / media items.

Interaction. The logic of the interaction is treated as integral part of authoring the narrative space. Each narrative object can have an interaction annotation which specifies the logical form, i.e. the variables and their values in each possible choice, the cues to engagers, the duration for which it should be active, i.e. how long the system should wait for engager interaction, and a default value, denoting what should be used in the absence of the engager's input.

Expressions. These are statements made with media items attribute names, interaction variables and context variables (see below). The can evaluate either to a Boolean value or to a set of media items. Expressions are used in the composition structures.

Link Structure. A link structure is a *directed graph*, possibly with cycles. Each node must be a narrative object. Each edge specifies a potential path that the narrative could take from the origin to the target node. Each edge has an enabling condition, which is a Boolean expression referring to the metadata of the narrative objects, input from the engager and context information (such as the play-list compiled so far). If the enabling condition of an edge originating in an object currently in the top of the playlist evaluates to true, the target node can immediately follow the origin in the playlist.

Branching out to more than one narrative object is done via *decision points*. One reason for introducing decision points as explicit objects is the need to disambiguate the situations where either more than one or no direct path is enabled. This is done by associating default rules the decision points.

Each link structure must have a start node. Interpreting a link structure involves starting at the start node and following enabled edges through the graph, until an end point (a point from which there are no links) is reached.

The following example[1]

```
link_structure(example_link,
  object(A, interaction(choice(name(x),item(short), item(long)))),
  start_item(object(A)),
  link(object(A),decision_point(D)),
  link(decision_point(D), object(B), enabling_condition(x=short)),
  link(decision_point(D), object(C), enabling_condition(x=long)),
  default(D, x=short))
```

specifies that an object A has attached a simple interaction as a choice between a long or a short play and is followed by a link to the short object (B) and the long object (C), with corresponding enabling conditions.

Layer Structure. A layer structure has a number of layers, each consisting of one narrative object. Interpreting a layer structure leads to the narrative objects referred to by each layer being added *in parallel* to the play-list, meaning that they or their content are "played" concurrently, starting at the same time. If the referred narrative objects are media items, then, indeed, they would end up being played in parallel. If they are not primitive, then they are further interpreted and replaced in the playlist with their corresponding (sub)playlists. Layer structures can be used, for example, to associate audio (soundtrack) with video.

Synchronisation between the content of the layers is expressed with a number of predefined actions, including: trigger, start, terminate, and end. Actions can be associated with the beginning and end of any narrative object and with any time point or time interval of an atomic narrative object. If an object that is selected for inclusion in the playlist has associated an action, then its associated action is interpreted (for simplicity this could also be read executed). For example if an object has associated with its beginning the action trigger:

```
object(A, action(name(T), type(trigger), time(beginning)))
```

then a signal called (T, trigger) is activated when object A is interpreted; the effect is that objects that were or would

[1] To improve readability, the NSL syntax has been slightly modified in all the examples in this paper.

become waiting for a signal T could process the trigger – its effect is to instruct to start the object waiting at the time when A is started. A listening object, say B, would have the following description:

```
object(B, listens(action(name(T)))
```

An example of a synchronisation between a link structure on one layer and an ANO on another one is given below:

```
layer_structure(example_layer,
  layer(1, link_structure(instance_link,
    start_item(object(A)),
    link(object(A),object(B, action(name(T), type(trigger), time(beginning)))),
  layer(2, ano(C, listens(action(name(T)))))
```

The object C on the second layer will be instructed to play only when object B on layer 1 starts to play.

Selection Group. The selection group is an extremely powerful structure. It may work with a set of explicitly selected objects or with objects selected via an expression. It may have pre-defined content, provided before delivery, or dynamic content, provided during delivery. It may refer to the choice of one object or a sequence of objects. Each of these will be discussed under a different heading.

Explicit Selection Group, or selection group with enumerated content. Such a group consists of content, selection criteria and termination condition. The content is a set of narrative objects specified via their ids. A group, when reached in the narrative space, is interpreted to one of the narrative objects it contains; the selected narrative object must satisfy the stated criteria. The selection criteria are specified with three predefined types of statements, which can be nested: selection rule, refinement rule and alternative rule. The selection filters the available objects based on some conditions. Then, if the resulting list consists of more than one object, the refinement rule is used, but on the selected objects only. If no objects are initially selected, the alternative rule is invoked on all the initially available objects. The following expression specifies a selection group with 5 objects, a selection criterion according to which there must be at least one topic description match between the selected object and the previously played object, a refinement that maximizes the number of matches and an alternative that selects randomly:

```
selection_group_structure(example_selection_group,
  objects(A, B, C, D, E)
  criteria(select(matches(description(this, class(topic)),
                          description(previous, class(topic))))
        refine(maximise(matches(description(this, class(topic)),
                                description(previous, class(topic))))),
        alternative(random)))
```

The above example uses a pre-defined context variable "previous" which refers to the immediately previously played object (see below).

A selection group may be looped, in which case it is interpreted until a termination condition becomes true.

Context Variables. NSL provides a number of pre-defined context variables, predicates and functions, including:

- *played*(Object) – true of the object has been played;
- *play_count*(Object) – the number of times the object has been played;
- *last_played_from*(Structured_Object) – last atomic object played from the structured object;
- *previous* – the narrative object interpreted previously that was at the same level (of depth in the narrative space) as the object in which "previous" is used,

and many ore. They can be used in any expressions, not just as selection criteria.

Filtered and Dynamic Selection Groups. The only difference between these and the explicit selection group is the way in which the content is defined. A filtered group selects its content from the whole pool of media items on the basis of a filtering expression which is evaluated at the time when the narrative is started, i.e. at each viewing. The expression is made with reference to metadata descriptions of the media items.

Filtered groups ensure the separation of the narrative space from the pool of media items. They allow the addition and/or removal of media items from the media pool between different deliveries. Further, if all the content is referred to via filtered groups, then the respective narrative space becomes a programme format that is reusable with different repositories of media items – i.e. a computational programme format.

Dynamic selection groups allow content to reach the narrative space during the delivery of the narrative. This is a powerful mechanism for dealing with user generated content. This is a hook to external procedures for inputting content, as NSL does not specify *how* new content could reach the narrative space.

Multi Select Groups. This is a structure that supports the specification of a sequence of objects in the playlist subject to a set of pre-defined constraints expressed with metadata attributes, input values and context variables. Examples include specifications of:

- minimum or maximum number of objects or total duration;
- that no object in the sequence should exceed a certain length;
- that an object satisfying a selection expression must or must not be included in the sequence;

The predefined constraints could also specify relationships between narrative objects within the scope of multi select groups. Such relationships refer mainly to the conditional existence or relative ordering, for example:

- one element from a set of objects should always be played before any object that is played from another set;
- two objects, if played, should be adjacent in the sequence (or a formulation with sets similar to the one above);

Currently, NSL does not distinguish between constraints and relationships – there are only constraints. However, we

are working on incorporating support for the specification of relationships between narrative objects and the employment of such relationships in the specification of the constraints. An example of a multi select structure is given below:

```
multi_select_group_structure(example_multi_select,
  filter(select(matches(character, list(Roope, Julia)) and
                matches(location, bar))
  multiselect(key_object(matches(character, Roope) and
                         matches(character_action, sings)) and
              length < 00:03:00:00 and length > 00:02:00:00 and
              must_follow(matches(character_action, love_song),
                          matches(character_action, kiss))))
```

It specifies that all the media items that have both characters Roope and Julia and for which the action is located in a bar should be included in the group. From them, a sequence of minimum 2 minutes and maximum 3 minutes should be compiled such that a key object that contains Roope singing should necessarily be selected and if an object in which there is a love song is selected, then it should necessarily be followed by an object in which there is a kiss.

Editing rules. Editing rules are similar to the constraints expressed in multi select groups, but should be able to be specified within the scope of any narrative object; obviously, a different but not disjoint set of predefined rules from those of multi select groups will be used for this purpose. This mechanism is not yet completely defined in NSL, as we still have issues regarding the selective applicability of certain rules to certain structures and also with recursive propagation to the objects within the object in whose scope the rule was specified and, subsequently, resolving possible collisions.

An example of a simple rule that may be specified at the level of the narrative space itself is:

```
rule(disallow(matches(copyrights, copyright_restricted)))
```

which specifies that copyright restricted material should not be used.

Conclusion. The recursive nature of the composition structures and their corresponding specification mechanisms make the core NSL a very powerful specification language. NSL was directly illustrated in this section. However, whereas it was intended to have a readable syntax, it was not intended to be used directly by authors. For that we implemented an authoring environment. The remainder of this paper provides a brief overview of the authoring environment.

Authoring Environment

The authoring environment supports the conception, development, and testing of interactive narratives. The following workflow processes are in its focus:

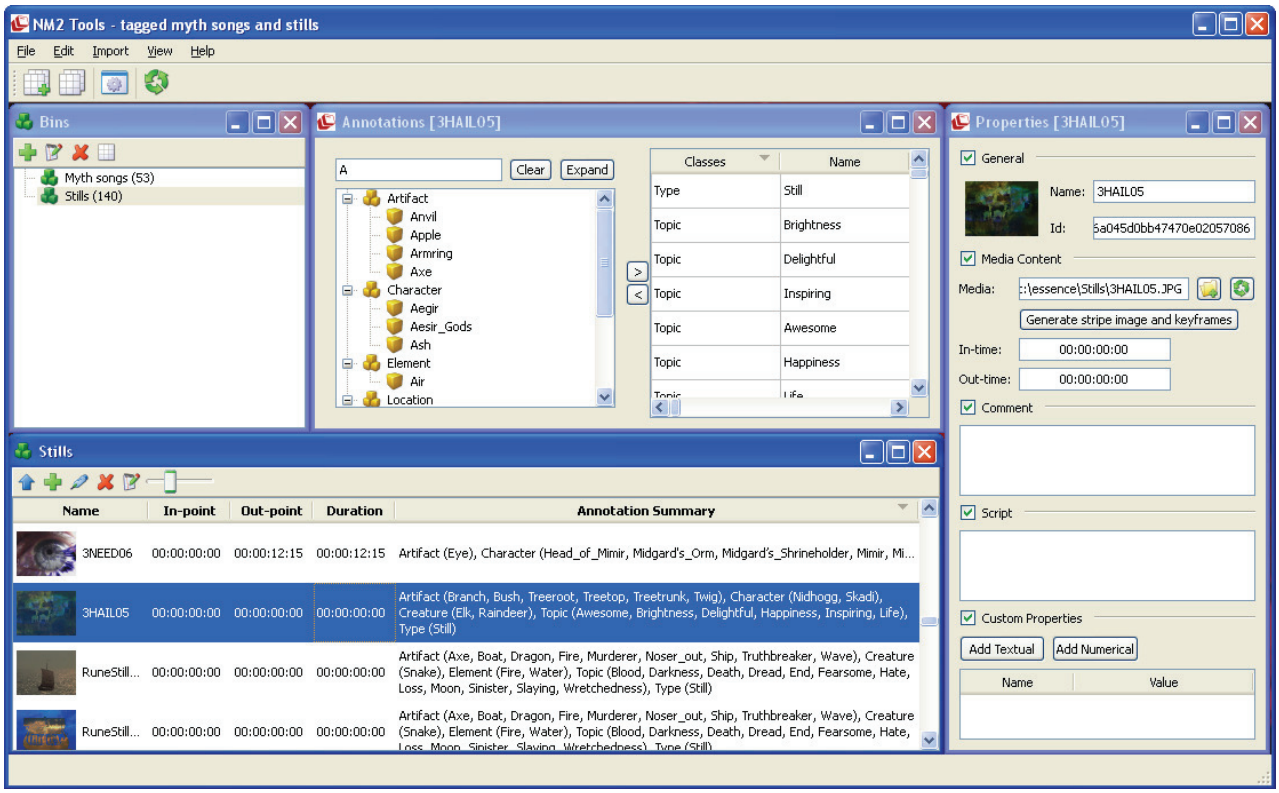

Figure 1. Media Item Description

- ingestion of pre-prepared media essence;
- description of ingested media (tagging with metadata);
- authoring of narrative space;
- testing of final production.

The Authoring Environment consists of a number of tools which could be combined in perspectives when carrying out the processes mentioned above – a perspective is an arrangement of a (sub)set of tools relevant to a certain process.

Ingestion is the process whereby the essence is brought into the system. The production of the media essence is outside the scope of our system, as it is already supported by a number of commercially available professional tools (such as Avid and Final Cut Pro). Ingestion results in the definition of the media items that will be used in the production.

Description is the process of associating metadata with the media items. The authoring environment supports the incorporation of plug-inns that carry out automatic feature detection and thus description. However, as many features required in the specification of the narrative space cannot be extracted reliably in an automatic fashion, the Authoring Environment provides tools for manual description (refer to Figure 1). Categories from a pre-edited ontology can be associated with each media item (the Annotation window). The production ontology can be altered at any point, but this feature is not illustrated. The media item view (top right) provides a quick overview of the collection of media items (which are organised in bins), and the property window (left bottom) allows the redefinition of the media items. There is a also a trimming tool which was not illustrated in this perspective.

The Authoring Canvas (refer to Figure 2) supports the development of narrative spaces. The canvas in Figure 2 illustrates a selection group narrative object which contains a number of a link structured narrative objects, which, in turn, contain both atomic and structured narrative objects (in their nodes). The top of the window specifies the context: the selection group is at a depth of 4 in the narrative space, within, in order, 1 link structure (root), one layer (entry), and another link (narrative). There is also a tree view of the narrative space. All the representations are synchronised and drag and drop and copy and paste operations are allowed between the tools. The Authoring Canvas enforces to a large extent the syntax of NSL.

Narrative spaces do not depend on the existence of high quality media. They can be constructed with placeholders or with representative media items of lower quality and/or cost (such as stills instead of video).

Currently we are in the process of implementing a GUI for authoring selection and editing rules, which, at present, are entered as text.

A number of mechanisms for testing and validating productions are supported in the authoring environment:

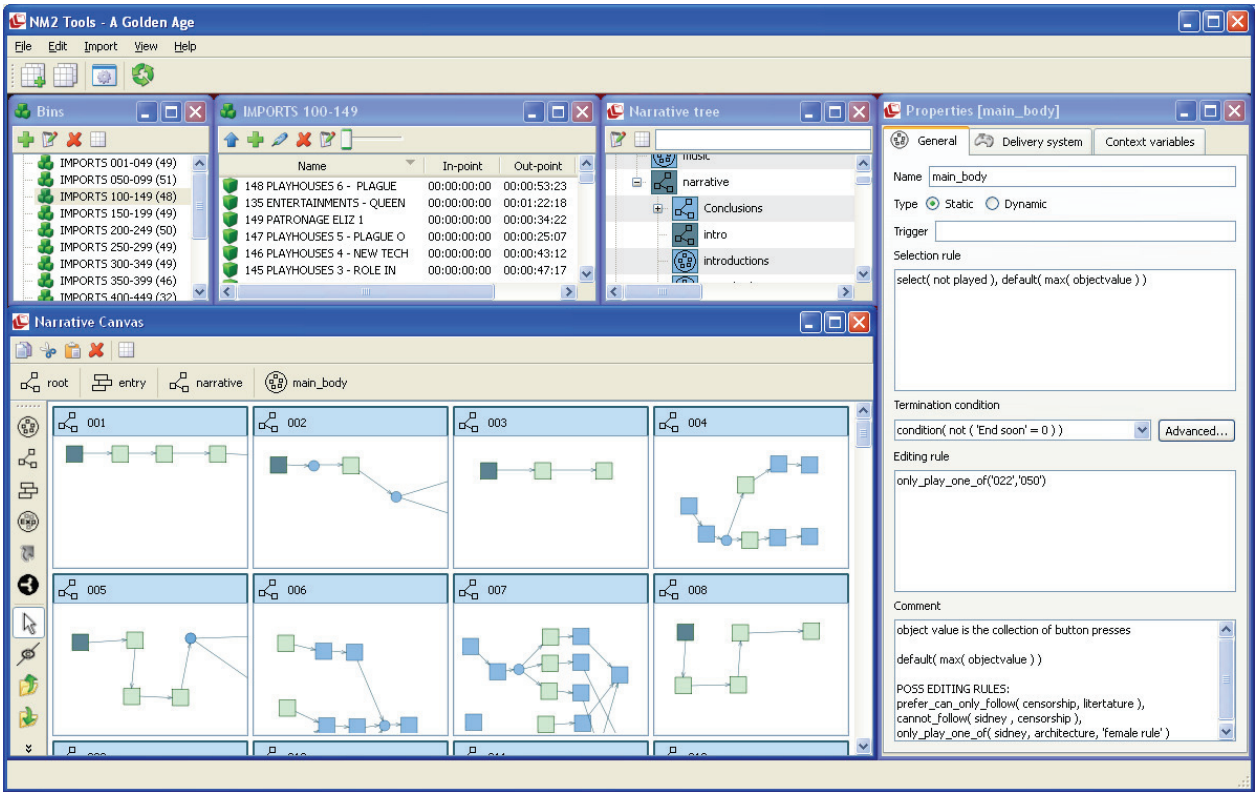

Figure 2. Narrative Space Authoring

syntactic checking, previewing and playlist viewing (refer to Figure 3). A generic interaction window abstracts the logic of the interaction and allows choices to be made for testing. The preview tool in conjunction with the interaction window supports the realisation of the overall narrative space and, to some extent, of any narrative object from the narrative space. This works very well for objects whose realisation does not depend on details (such as user input) provided in other objects. We are currently working on a general solution at this end, whereby any narrative object from the canvas could be visualised (currently, we are parsing form the start object of the narrative space and employ default choices until we reach the object to be previewed). Crucial for testing is the playlist viewer. This shows what is being played in the preview tool, gives details about it and provides an easy access to the place in the narrative space (on the canvas) where the media object originates from.

The Authoring Environment in its present incarnation is not "intelligent". However, we have immediate plans of incorporating semantic tests which will implement heuristics by which warnings and advice could be given to authors regarding the structures they build.

Conclusions

We have developed means for *thinking about*, *discussing*, *authoring*, *testing* and *delivering* interactive screen media productions. Our approach and the accompanying system have been validated via a number of productions, notably through the dramatic production Accidental Lovers. To our knowledge, we are the first to have developed a *production and genre independent* computational model and associated system that was employed in the realisation of a truly interactive production delivered on a national television channel. We believe that our approach will continue to be used and will gradually lead to the refinement of a richer language of interactive narratives, which would capture elements of grammar and rhetoric of the newly emergent medium.

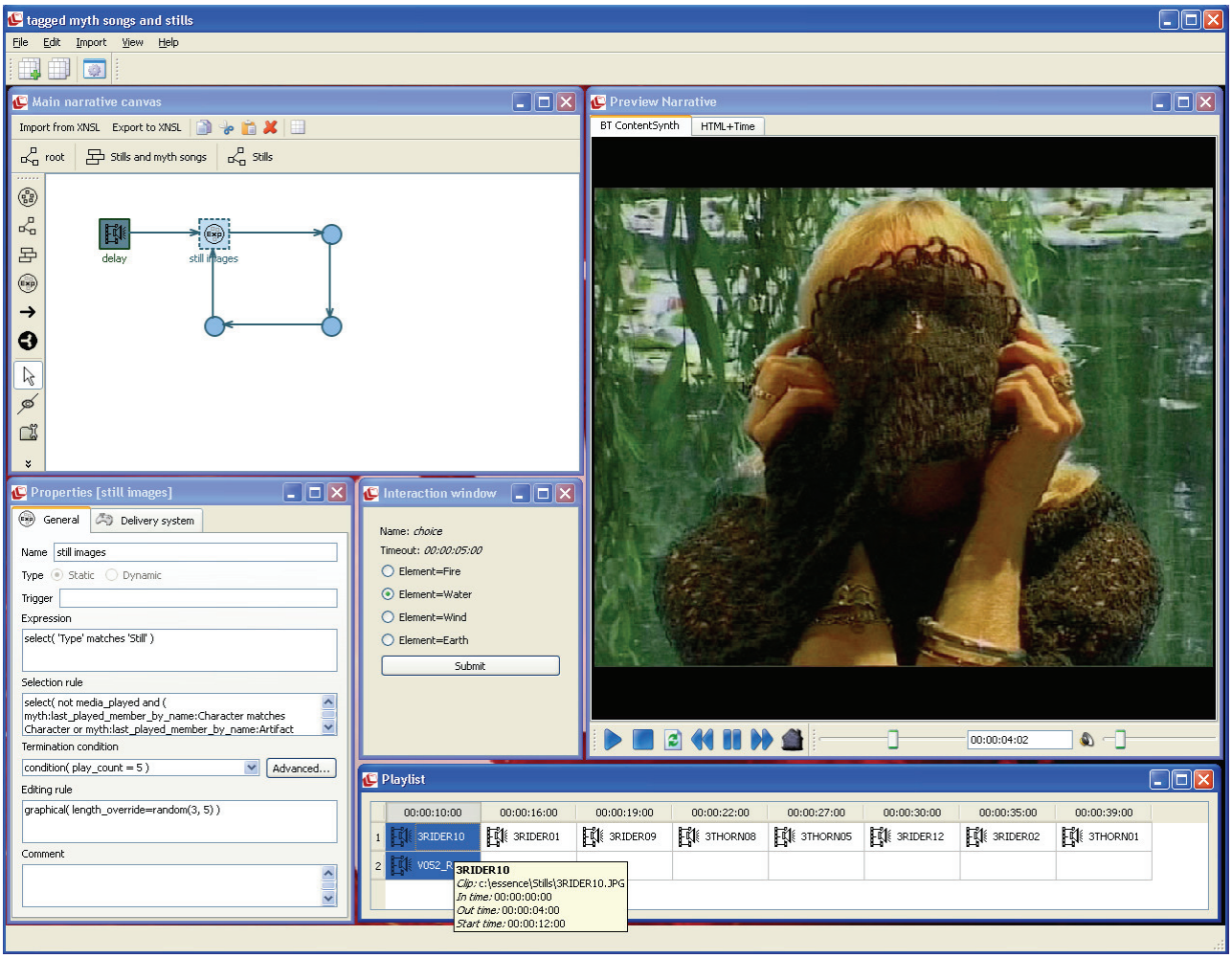

Figure 3. Preview

There are a number of directions we are considering for further developments:

- refine higher level structures for NSL, possibly genre specific, that are closer to the author's thinking than to the mechanics of the composition, and which will be compiled down to the core structures of NSL presented here;
- implement editing rules on the formal basis of a deontic logic;
- implement a comprehensive set of semantic tests that could be employed in providing intelligent support for authoring, testing, and validation;
- refine a richer authoring knowledge base and mechanisms to apply such knowledge during authoring – to constrain, advise or even carry out more routine tasks of authoring – and during the compilation of narrative threads;
- develop a more powerful optimizer; this is strongly linked to the previous aim.

We believe that the computational model and the system we developed represent a solid foundation for these developments.

References

Agmanolis, S.; and Boye Jr., M. 2003. Viper: A Framework for Responsive Television. *IEEE Multimedia*, 10 (1): 88-98.

Barry, B. 2005. Mindful Documentary. PhD Thesis at the Media Lab MIT, Massachusetts, USA.

Bulterman, D.C.A.; and Hardman, L. 2005. Structured Multimedia Authoring. *ACM Transactions on Multimedia Computing, Communications and Applications* 1(1): 89-109.

Cavazza, M.; Charles, F.; and Mead, S.J. 2002. Character-Based Interactive Storytelling. *IEEE Intelligent Systems*, Special Issue on AI in Interactive Entertainment: 17-24.

Cavazza, M.; and Charles, F. 2005. Dialogue Generation in Character-Based Interactive Storytelling. In *Proceedings of AAAI First Annual Artificial Intelligence and Interactive Digital Entertainment Conference*, Marina del Rey, California, USA. AAI Press.

Gordon, A.; van Lent, M.; van Velsen, M.; Carpenter, M.; and Jhala, A.: Branching Storylines in Virtual Reality Environments for Leadership Development. In *Proceedings of the Sixteenth Innovative Applications of Artificial Intelligence Conference (IAAI-04)*, San Jose, CA. Menlo Park, California, USA. AAAI Press.

Mateas, M.; and Sengers, P. 1999. Narrative Intelligence: An Introduction to the NI Symposium. *In the Working notes of the Narrative Intelligence Symposium*, *AAAI Fall Symposium Series*. Menlo Park: Calif.: AAAI Press.

Mateas, M.; Vanouse, P.; and Domike S. 2000. Generation of Ideologically-Based Historical Documentaries. In *Proceedings of AAAI 2000*, Austin, TX: 36-42

Mateas, M.; and Stern, A. 2005. Structuring Content in the Façade Interactive Drama Architecture. In *Proceedings of AAAI First Annual Artificial Intelligence and Interactive Digital Entertainment Conference*, Marina del Rey, California, USA. AAAI Press.

Riedl, M.; and Young, M. R. 2006. From Linear Story Generation to Branching Story Graphs. *IEEE Journal of Computer Graphics and Applications*, 26(3): 23- 31.

Tuomola, M.L. (director); Saarinen, L.E (writer); and Nurminen, M.J. 2006/2007. Accidental Lovers ("Sydän kierroksella"), Crucible Studio, Helsinki University of Art and Design Finland. Broadcast on Channel TV1 by YLE, The Finnish Broadcasting Company.

Ursu, M.F.; Cook, J.J.; Zsombori, V., Zimmer, R.; Kegel, I.; Williams, D.; Thomas, M.; Wyver, J.; and Mayer, H. 2007a. Conceiving ShapeShifting TV: A Computational Language for Truly-Interactive TV. In P. Cesar et al. (Eds): *EuroITV 2007*, *LNCS 4471*: 96 - 106.

Ursu, M.F.; Thomas, M.; Tuomola M.L.; Wright, T.; and Zsombori, V. 2007b. Interactivity and Narrativity in Screen Media. Forthcoming in the *Proceedings of the IEEE Symposium on Multimedia Systems*, Taichung, Taiwan, R.O.C, December 10-12, 2007.

Young, M.R. 2001. An Overview of the Mimesis Architecture: Integrating Intelligent Narrative Control into an Existing Gaming Environment, in *The Working Notes of the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment*, Stanford, CA, March 2001.

Wages, R.; Grützmacher, B.; and Conrad, S. 2004. Learning from the Movie Industry: Adapting Production Processes for Storytelling in VR. In Göbel, S., Spierling, U., Hoffmann, A., et al. (eds.) *Technologies for Interactive Digital Storytelling and Entertainment: Second International Conference, TIDSE 2004*, *Darmstadt, Germany, LNCS 3105*, Springer-Verlag: 119 - 125.