H. Reiterer & O. Deussen (Hrsg.): Mensch & Computer 2012
München: Oldenbourg Verlag, 2012, S. 409-418

# Calliope-d: An Authoring Environment for Interactive Dialog

Robert Walter, Katja Neuwald

Entertainment Computing Group, University of Duisburg-Essen

**Abstract**

Creating an interactive dialog for a video game is a complex task. Unlike in non-interactive media, the requirements on a game dialog are unique in many ways and writers have to reinterpret their role in almost every project. This not only addresses the artistic challenge of writing high quality dialog lines, but also the preservation of consistency throughout every possible path, the co-ordination with the game's pace, and its technical integration into a game engine. Surprisingly, there is only little professional tool support for game writers so far. In this paper, we introduce the authoring tool Calliope-d for the creation of interactive dialog. It builds upon a series of interviews with professional game writers. The main feature of the prototype is the separation of the complex structure of an interactive dialog and its actual content. We will show the benefits of this approach and how the software design allows for an arbitrary creation of abstractions, even for the integration of dialog in game engines.

## 1 Introduction

Digital games share many similarities with other media, but at the same time, games are different. As described by Ara Shirinian, games are undeniably good at two things: First of all, they are good at providing gameplay, something that no other modern medium is capable of. Secondly, digital games have an immense potential in terms of visual narration, as they have almost "all the same capabilities that film does" (Shirinian 2010), but allow for another depth of involvement because of their unique interactive nature. This additional dimension of complexity entails manifold consequences for the creation of a game narrative.

Players constantly act and make choices, and they demand games where their decisions actively influence their hero's path. Hence, writers always have to keep in mind that players are not pure recipients (like in a film or a novel), but discoverers or even creators, as has been described by Sweetser (Sweetser 2007). Thereby players primarily want to play a game. No matter how great a game's story is, if the gameplay does not work, neither will the game, so story always has to follow gameplay (McDevitt 2010). As a consequence, story-driven games feature complex narrative structures which need to be thoroughly woven into the

gameplay. This is especially true for interactive dialog. Compared to cut scenes, in-game artifacts, or scripted in-game events, dialog is interactive by nature and can rapidly grow to complex structures, which makes it hard to maintain consistency throughout every possible path. Thereby, a dialog can be strongly intertwined with a game's current state on an extremely fine-grained level. This means that the flow of a dialog may not only depend on what option the player is choosing directly, but also on which information has been gathered so far, which dungeon has been mastered, or which items have been found.

An obvious approach towards authoring tools that support the creation of interactive dialog is an editor that features a graphical representation of the dialog structure. However, game writers – often having a background as screenwriters or novelists – are not familiar with such editors and the computer-aided creation of tree structures or state charts. This makes it hard for them to use such interfaces. In a series of interviews we did with professional game writers and designers like Ron Gilbert, Richard Dansky, Steve Ince, Falko Löffler, or Kevin Mentz, we received the uniform feedback that, although graphical representations might be a plus to keep track of a narrative flow, writers in the first place need to be able to focus on the word, the actual content they have to create. In Table 1, we compiled some of their statements.

| **Interviewee** | **Statement** |
|---|---|
| Ron Gilbert | I don't use visualizations of dialogs, no. |
| Richard Dansky | I preferably use script format. |
| Steve Ince | In 17 years of professional game writing, I've never created a dialog tree. |

*Table 1: Sample statements of game writers and designers regarding the use of visualizations for creating branching dialogs*

A recent survey on storytelling tools (Rabil 2012) underlines this, as one participant explicitly stated "When I create stories, high level design documents and write dialogue, I want to be able to do so without the interface getting in the way. I want to do as much as possible through the keyboard and only reach for the mouse when absolutely necessary", while another one asked "that the bells and whistles don't overwhelm the actual writing element of the software. It should be as easy to use as Final Draft or Word for writing."

In this paper, we introduce the dialog authoring prototype *Calliope-d*, the first implementation in the context of the Calliope project. The Calliope project aims at defining a high-level concept that describes how to create game writing tools in general, providing both domain-specific requirements as well as a dedicated software design. In this first instance, we created a hybrid dialog authoring environment combining textual as well as graphical user interfaces of branching dialog to address the above-described demands. The different views represent a separation of concerns: While the textual view focusses on the creation of the dialog (both its content and its structure!), the graphical view makes the structure explicit and provides important meta-information on every dialog node. Another kind of view is the export in XML for the integration of the dialogs into a game engine. Using a MVVM (Model-View-

ViewModel, a specialization of the Presentation Model by Martin Fowler (Fowler 2004)) architecture makes the user interface independent from the underlying data model, which allows for the creation of arbitrary views in the future.

# 2 State of the Art

The creation of authoring tools for interactive dialog is a major concern in both the research field of Interactive Storytelling as well as most recently the gaming industry. This chapter provides a selection of approaches and tools to illustrate the achievements so far.

## 2.1 Research Approaches

The IRIS (Integrating Research in Interactive Storytelling) network of excellence[1] is a research initiative that heavily contributed to the research field of Interactive Storytelling. In the Scenejo project (Spierling et al. 2006), members of the IRIS network work on a platform that allows writers to model and evaluate conversations with a transition graph representation. The created dialog is written for a chatbot-based environment resembling the dialog-driven interactive drama Façade (Mateas & Stern 2005).

Another research approach towards authoring tools for interactive dialog is Swat by Chris Crawford, which is part of the Storytron environment (arisen from the Erasmatron platform of 1998). Swat allows to author so-called storyworlds for the runtime environment.[2] The Java-based system aims at a modular creation of stories by modeling the actions, emotions, and inclinations of virtual characters to create emergent narratives. Given the recent statements of Chris Crawford regarding the project, it is at least doubtful if the system will accomplish the defined goals.

StoryTec (Göbel et al. 2008) is a system to create interactive applications with a focus on rapid prototyping. The system's application field is manifold. It can be used to prototype story-based museum guides, web-based training and e-learning systems, simulation and training environments, and exergames.

ScriptEase (McNaughton 2004) is a "visual tool for the creation of computer role-playing games". The approach is to provide different kinds of patterns that enable the creation of complex behaviors on a high level of abstraction. The tool provides four kinds of patterns, namely encounter, behavioral, dialog, and plot patterns, which have been derived from the game Neverwinter Nights as the four main patterns in computer role-playing games. Users of ScriptEase can create logical relations between objects, characters, and events using their generative design patterns. ScriptEase then generates script files in the proprietary NWScript

1 http://cordis.europa.eu/fp7/ict/content-knowledge/iris_en.html

2 Note: The web references for the Swat system are unavailable during the creation of this contribution. Please find a wiki of the Storytron system here: http://www.ifwiki.org/index.php/Storytron

format (the scripting language of the Aurora toolset, see the following subsection, based on the language C). These scripts can then be run within the Aurora runtime.

In comparison to Calliope-d, the formerly mentioned systems allow the creation of proprietary story or dialog scenarios which are executable in a corresponding runtime. They all focus on graphical user interfaces to model interactive relations of story events. Although the usability and an appropriate target group orientation is a concern of some of the systems, none of them provide a dedicated abstraction that allows the creation of platform-independent, interactive dialog. Calliope-d features a sophisticated and at the same time highly abstracted representation for interactive dialog to address game writers' requirement of simplicity. In other words, the content-wise creation of interactive dialog in Calliope-d is completely decoupled from technical systems and allows for a target group-oriented process, which is meant to be included into actual game development processes.

## 2.2 Professional Authoring Tools for Interactive Dialog

Dominant solutions for the creation of an interactive narrative and specifically interactive dialog are makeshift solutions like spreadsheets and screenwriting tools (Despain 2008) or proprietary scripting languages (Ince 2006). Spreadsheets and screenwriting formats leave a void between the created narrative content and its integration into the game since they do not provide any domain-specific semantics. This makes a reasonable iterative development process very hard and the creation of the narrative very error-prone.

Scripting languages require at least basic programming skills, which game writers more than often do not have. Often working as freelancers for different studios requires them to learn several Excel formats and scripting languages, and game writers eventually are more occupied in complying with a proprietary format than developing a good narrative.

Other proprietary solutions are toolsets built for specific games, which mostly are made public only in case that the developer wants to enable the end-users to create their own stories within their game environment. Popular examples of that approach are the Aurora and Electron toolsets, developed for the game Neverwinter Nights (Aurora, developed by BioWare) respectively Neverwinter Nights 2 (Electron, developed by Obsidian Entertainment). Although these toolsets feature some interesting abstractions for the creation of a game story, their proprietary nature limits their application in a professional context.

As a first commercial tool for creating branching dialog, there is Chat Mapper (Urban Brain Studios). Chat Mapper is built by industry professionals "from the frustration of trying to use Final Draft, Word, or even Excel to write complex branching dialogue […] from the ground up with usability in mind."[3] What centers the tool is a branching tree graph where the writers have to create their conversations. Moreover, Chat Mapper offers a wide range of export formats, namely JPEG, PNG, Word, Excel, and XML. Custom exporters can be implemented

[3] http://www.chat-mapper.com/

using the featured SDK. Chat Mapper also contains a simulation environment which enables writers to test run their dialogs.

Articy:draft (Nevigo GmbH)[4] is another commercial tool, which not only focusses on the creation of interactive dialog, but on characters, objects, worlds, and the story itself. Similar to ChatMapper, articy:draft features a structural visual representation of dialog networks, which is called flow editor.

The professional tools represented here solely rely on a graphical representation for interactive dialog. The creation of dialog using a graphical interface forces the author to frequently fulfill tedious tasks like create and arrange nodes, connecting them, drag and drop meta-information, or opening and navigating through properties views.

While Calliope-d also features some of these tasks, the main focus for the creation of interactive dialog lies upon the simple yet powerful textual editor, as we will see in the following chapter.

# 3 The Calliope Project

The Calliope project is intended to describe a framework for the feasible creation of game writing tools. Instead of a Swiss army knife approach for one generic game writing tool, our concept abstracts from realization issues to provide a catalog of architectural as well as target group-oriented best practices for the creation of game writing tools. The intention is that others can address the needs of their specific problem domain by conforming to our concept.

In order to be able to constantly verify our concept, we implement authoring prototypes that instantiate the best practices of the concept. The first result of this work is manifested in the dialog authoring prototype Calliope-d. Our central goal was to define a textual user interface that allows writers to solely focus on the creation of the dialog content. However, they should also be able to define the dialog's structure without getting distracted by its visualization. We achieve this by a screenwriting-like textual interface that features context sensitive buttons to locally decide how a dialog proceeds.

## 3.1 A Scripting Format for Branching Dialog

We identified two contradictory central requirements for interactive dialogs. On the one hand, authors want to have simple, script-like user interfaces that are easily accessible and allow them to focus on the text they have to write (compare Chapter 1). On the other hand, an interactive dialog has a graph-like structure by definition, which means that authors need to be able to decide which of the written dialog lines is a response or an alternative to a given line.

[4] http://nevigo.com

In order to solve this problem, we let the writer decide locally how a dialog is structured, knowingly omitting an explicit representation of the overall structure. For that, we use context-sensitive buttons, as can be seen in Figure 1.

The textual view is reduced to a simple paper metaphor. It provides some basic text formats for the project's header ("Branching Dialogue – Heavy Rain") and the dialog's header ("Lauren Winter & Scott Shelby"). Calliope-d allows the simple definition of characters, which can then participate in conversations. In the example, the characters Scott Shelby (the player character, PC) and Lauren Winter (a non-playable character, NPC) are having a conversation. For each dialog line, the writer has to select the speaking character from a drop-down menu (the name can also be typed manually; in case of a typing error, the writer is notified). What follows is the spoken dialog line, e.g. "The killer is walking around free as we speak […]". The writer then gets the opportunity to add one of three things: Either an *answer*, which is defined as a dialog line spoken by another character in direct response to the given line, or an *alternative*, which is defined as an alternative line to the given one, spoken by the same NPC, or an *option*, which is defined as an optional line to the given one, spoken by the same PC. Every dialog line is marked by in identifier. Below each dialog line, the writer can jump to the directly following dialog lines using the hyperlinks (e.g. "To: 2b"). We additionally utilized a post-it metaphor to enable writers tagging single dialog lines with important meta-information, for example for the audio recording.

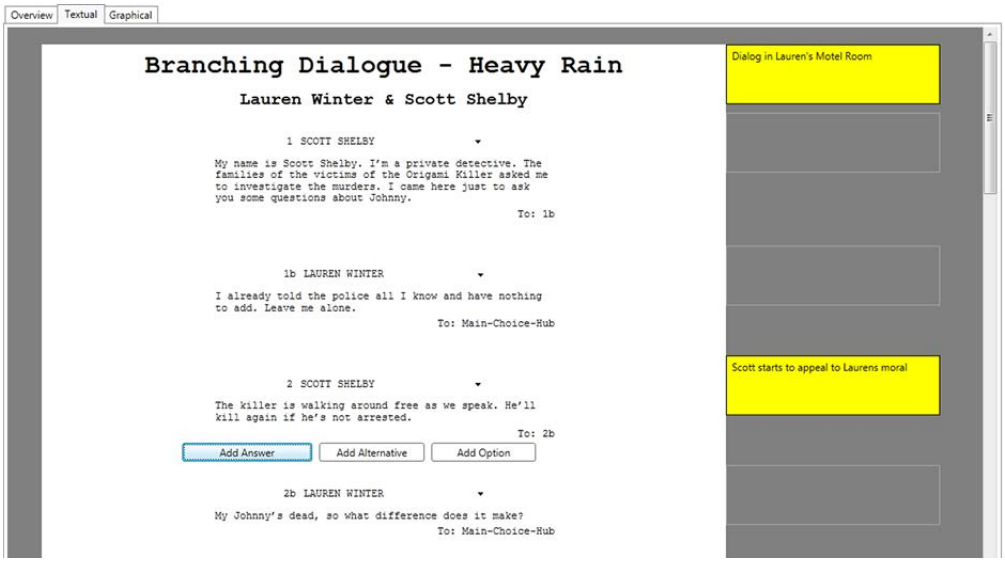


*Figure 1: Creating a branching dialog within the textual view of Calliope-d*

Thus, writers can create a branching dialog structure within a textual view, making *local* decisions on the dialog's flow. The dialog is stored in an underlying domain model, and the textual view is only one possible representation of that model. This means that it is possible to discuss, maintain and even enrich the dialog within a graphical view. Figure 2 shows that

graphical view for the example used in Figure 1. At the current state of Calliope-d, the graphical view is considered as a proof of concept to illustrate that the dialog created in the textual view actually features a structure which is stored in conformation to a common underlying data structure.

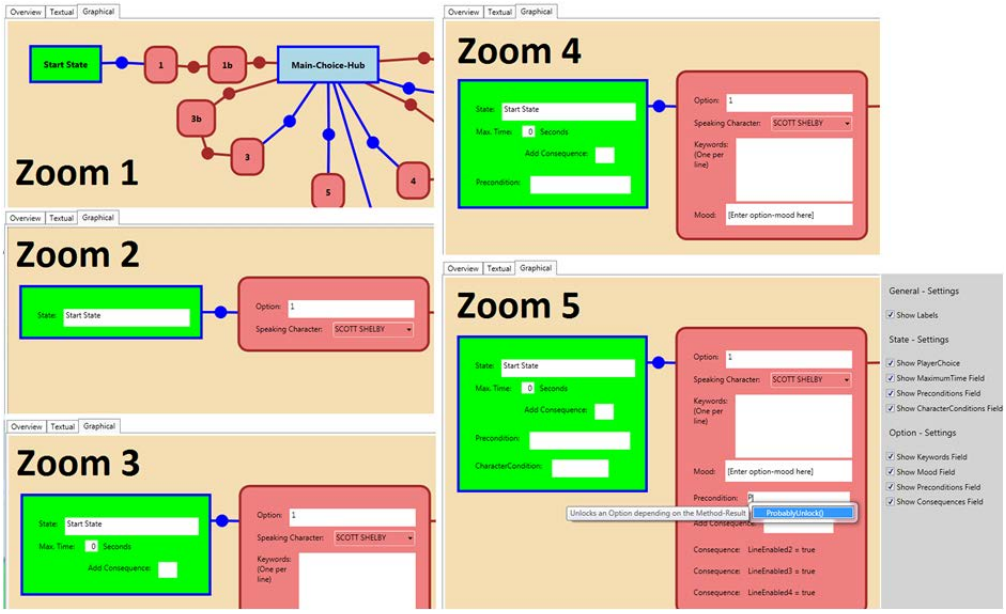


*Figure 2: Graphical views, showing different zoom steps*

While the first zoom step gives an overview of the dialog structure by merely displaying the identifiers of every line and their relation, starting with the second zoom step meta-information of every node is provided. The rectangular nodes represent explicit states of a dialog, which help visualizing branches. Dialog lines are displayed by rounded nodes. Outgoing state relations are blue, while outgoing dialog relations are red. Using the setting menu as shown for zoom depth five, it is always possible to hide and show the details of the dialog states and lines separately.

This separation of dialog content and its overall structure allows writers to focus on the creation of the dialog itself rather than struggling with graphical user interfaces when creating the dialog in the textual view. As we will see in more detail in the following section, all views rely on the same data model, so that changes made in one view are automatically distributed to all other views.

## 3.2 The Architecture of Calliope-d

What centers the Calliope concept and thus the Calliope-d prototype is the idea of an underlying model that formally describes a specific problem domain. This follows the philosophy

of Domain-Driven Design (DDD) as described by Eric Evans (Evans 2003). Working together with domain experts should lead to a common understanding of the domain's entities and their semantics. This means that domain experts should take an integral part in the creation of tools like Calliope-d to ensure that domain-specific concepts, like a certain terminology, are incorporated in the software's design explicitly. In Calliope-d, we established a ubiquitous language (as described by Evans) in cooperation with professional game writers and designers (see acknowledgements) that, for example, differentiates between answers, alternatives, and options as possible kinds of dialog lines. This differentiation is incorporated in the software design and the user interface, as has been described in Section 3.1. Together with other domain semantics, a *domain model* was derived. The domain model for Calliope-d eventually consists of four components: (1) the dialog structure that enables an arbitrary graph structure of dialogs; (2) the actual dialog content; (3) conditions & consequences to control the flow of the dialog during runtime; (4) descriptive information like character descriptions, mood descriptions, and free annotations.

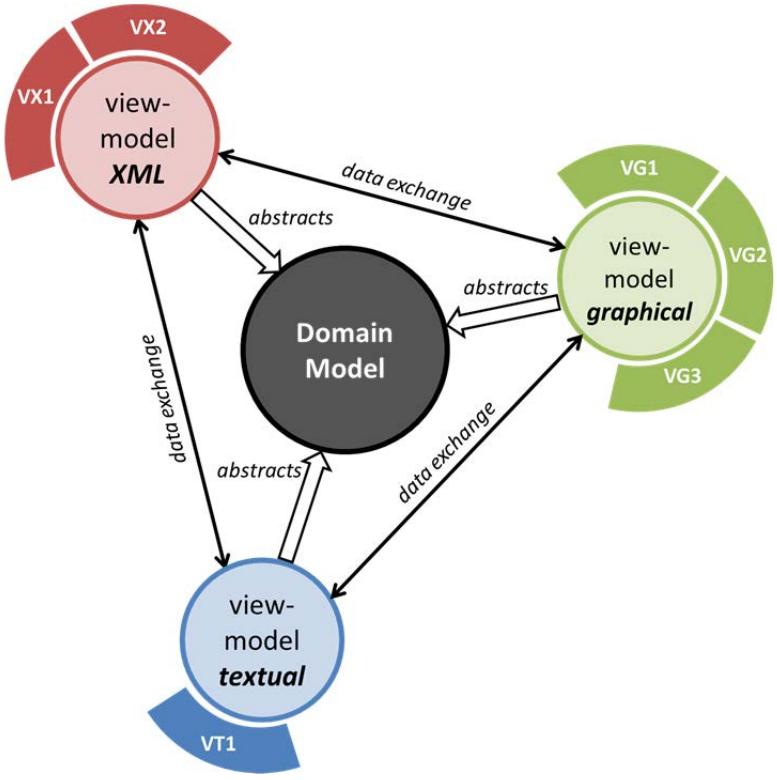


*Figure 3: Exemplary MVVM architecture of Calliope-d*

Based upon this domain model, Calliope-d features several abstractions, the so-called *view-models*. A view-model comprises the relevant data of the domain model and prepares the

data for its representation in the corresponding *views*. Hence, a view can be described as a task and target group-oriented representation of the view-model.

The architecture of Calliope-d is illustrated in Figure 3. There exist three different abstractions (the view-models for the textual view, the graphical views, and the XML exports) of the domain model, each with their individual views. Characteristically for an MVVM architecture, every change made in one view is instantly broadcasted to all other views if necessary, as indicated by the direct "data exchange" relations in Figure 3.

# 4 Conclusion

Calliope-d is the first prototype of the Calliope project. It features a domain-driven software design, realized in an MVVM architecture. This prototype illustrates the usage of such an approach and how the separation of different concerns can be mapped to a separation of view-models and their views. This brings benefits regarding the accessibility of the tool as well as the integration of written dialogs into a game engine. Defining a ubiquitous language with domain experts that manifests in the software design and the software's user interface ensures that the domain experts can use familiar semantics. Using a common domain model also allows the transformation of the data to different formats. In Calliope-d, the dialogs are exported to a proprietary XML format which can be utilized for game engines with a corresponding software API.

We hope that we can generalize this approach in our future work to make it more accessible for others. In order to do so, we are about to create a catalog of best practices for game writing tools in general, not only for the authoring of interactive dialogs. Moreover, we want to include some basic text editing features like cut, copy, and paste in order to let writers evaluate Calliope-d and its approach of separating textual and graphical representations.

Other parts we are about to include in Calliope-d are:

- proper logic editors that allow writers to add conditions and consequences for every dialog line using domain-specific languages,
- project-specific configuration, so that programmers can define constraints that are handled by the tool,
- ubiquitous context information, letting the writers access context information like concept arts or sound files while preserving the need for simple and accessible user interfaces, and
- more export formats, to enable additional exports, like actor scripts or Excel files.

For the textual view, we consider to add an optional, simple yet powerful color coding, which highlights the direct siblings of the currently marked dialog line.

**Acknowledgments**

We thank all interview partners for their time and insight into their daily work and business. We especially thank Ron Gilbert, Richard Dansky, Steve Ince, Kevin Mentz, Falko Löffler, Georg Struck, Jim Murdoch, Bernie Duffy, Daniel Dumont, Sven Hammer, and Christoph Brosius. Thanks to your input, we are able to research on and create systems like Calliope-d.

**References**

Despain, W. (2008). Interactive Script Formatting. In Despain, W. (Ed.): *Professional Techniques for Video Game Writing*. A K Peters.

Evans, E. (2003). *Domain-Driven Design: Tackling Complexity in the Heart of Software.* Addison-Wesley Longman, Amsterdam, The Netherlands.

Fowler, M. (2004). *Presentation Model*. http://martinfowler.com/eaaDev/PresentationModel.html.

Göbel, S., Salvatore, L., Konrad, R., and Mehm, F. (2008). StoryTec: A Digital Storytelling Platform for the Authoring and Experiencing of Interactive and Non-linear Stories. In *Proceedings of the 1st Joint International Conference on Interactive Digital Storytelling*, Erfurt, Germany.

Ince, S. (2006). *Writing for Video Games*. Methuen Drama.

Mateas, M., Stern, A. (2005). *Procedural Authorship: A Case-Study of the Interactive Drama Façade*. Digital Arts and Culture (DAC).

McDevitt, D. (2010). *A Practical Guide to Game Writing*. Gamasutra Article: http://www.gamasutra.com/view/feature/6171/a_practical_guide_to_game_writing.php.

McNaughton, M. et al. (2004). ScriptEase: Generating Scripting Code for Computer Role-Playing Games. In *Proceedings of the 19th IEEE International Conference on Automated Software Engineering.* Washington D.C., USA.

Rabil, R. (2012). *An Evaluation of Software Tools for Interactive Storytelling*. Issuu Document: http://issuu.com/richardrabiljr./docs/softwareevaluation_interactivestorytelling.

Shirinian, A. (2010). *The Uneasy Merging of Narrative and Gameplay*. Gamasutra Article: http://www.gamasutra.com/view/feature/4253/the_uneasy_merging_of_narrative_.php.

Spierling, U., Weiß, S., Müller, W. (2006). Towards Accessible Authoring Tools for Interactive Storytelling. In *Proceedings of the 3rd International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, Heidelberg, Germany.

Sweetser, P. (2007). *Emergence in Games*. Cengage Learning EMEA.

**Contact Information**

Robert Walter and Katja Neuwald
University of Duisburg-Essen
Forsthausweg 2
47057 Duisburg
robert.walter@uni-due.de , katja.neuwald@stud.uni-due.de