D 6.4

ANALYSIS OF INTERACTIVE STORYTELLING APPLICATIONS

Project Number	FP7-ICT-231824
Project Title	Integrating Research in Interactive Storytelling (NoE)
Deliverable Number	D 6.4
Title of Deliverable	Second Version of Interaction Framework[1]
Workpackage No. and Title	WP6 - Interaction and Dialogue
Workpackage Leader	UOA
Deliverable Nature	Report
Dissemination Level	Public
Status	Final
Contractual Delivery Date	30th June 2011
Actual Delivery Date	23rd July 2011
Author(s) / Contributor(s)	Gregor Mehlmann (UOA), Felix Kistler (UOA), Birgit Endrass (UOA), Elisabeth André (UOA), Christoph Klimmt (HMTM), Johannes Wagner (UOA), Remi Smirra (UOA)
Number of Pages	35

[1] Original title: Second Version of the Conceptual Framework. The title has been changed in favor of greater concreteness.


IRIS

Table of Contents

Abstract

In this Deliverable, we describe an updated version of the technical framework presented in Deliverable D6.3. The major extensions include:

- For an easy and fast creation of the domain knowledge used by the natural language understanding engine we implemented a graphical editor tool. This tool allows a faster creation and modification of the dialogue domain knowledge and a direct testing of the engine's classification quality.
- For natural full body interaction based on Microsoft Kinect, we came up with taxonomy of gestures and postures. For each category, a repertoire of sample gestures and postures were implemented.

In addition, we conducted two new studies to investigate different style of interaction:

- The first study focused on full body interaction using Microsoft Kinect and compared interaction based on natural gestures with interaction based on gestures emulating a computer mouse.
- The second study focused on language-based interaction and compared round-based vs. continuous interaction.

1. TECHNICAL INTERACTION FRAMEWORK

In this section, we describe the extensions of the technical interaction framework presented in Deliverable D6.3. The technical interaction framework is based on a conceptual analysis of existing interactive story telling systems. Our focus is on the provision of communication means common in human-human conversation, such as speech, posture, gestures and gaze. First of all, these communication means matched the style of our IS scenarios best. Secondly, we aimed to keep the time required to get acquainted with an IS at a minimum.

As a consequence, the technical interaction framework supports the following modes of interaction:

- Language-based interaction (text and speech)
- Full body interaction (gestures and poses)
- Gaze-based interaction
- Direct manipulation

In addition, there is a component for handling multimodal multi-threaded dialogues between the virtual story characters and the user.

Since the most significant changes after publishing deliverable D6.3 have been performed for language-based and full body interaction as well as dialogue management, the current deliverable focuses on these components.

1.1 LANGUAGE-BASED INTERACTION

1.1.1 Dialogue Management with the Scenemaker Modelling Tool

Central concept of Scenemaker's modeling approach is the separation of dialogue content and structure. Multimodal dialogue content is specified in a set of *scenes* that are organized in a *scenescript*. The narrative structure of an interactive performance and the interactive behavior of the virtual characters is controlled by a *sceneflow*, which is a state chart variant specifying the logic according to which scenes are played back and commands are executed. Sceneflows have concepts for the *hierarchical refinement* and the *parallel decomposition* of the model as well as an exhaustive *runtime history* and multiple *interaction policies*. Thus, sceneflows adopt and extend concepts that can be found in similar state chart variants (see (Harel, 1987) and (Beeck, 1994)). Sceneflows and scenescripts are created using a graphical modeling tool and executed by interpreter software. This allows the real-time extension and modification of the model and the direct observation of the effects without the need for an intermediate translation step or even the need to pause the execution of a sceneflow. The real-time visualization of a sceneflow's execution and active scenes within the graphical user interface allows testing, simulating and debugging the model. These features facilitate and accelerate the modeling process and, thus, allow the creation of interactive virtual character performances in a rapid-prototyping style.

1.1.1.1 Creating Multimodal Dialogue Content

As shown in Figure 1, a scene resembles the part of a movie script consisting of the virtual characters' utterances containing stage directions for controlling gestures, postures, gaze and facial expressions as well as control commands for arbitrary actions realizable by the respective character animation engine or by other external modules. Scenescript content can be created both manually by an author and automatically by external generation modules.

The possibility to parameterize scenes may be exploited to create scenes in a hybrid way between fixed authored scene content and variable content (Figure 1 (1)), such as retrieved information from user interactions, sensor input or generated content from knowledge bases.

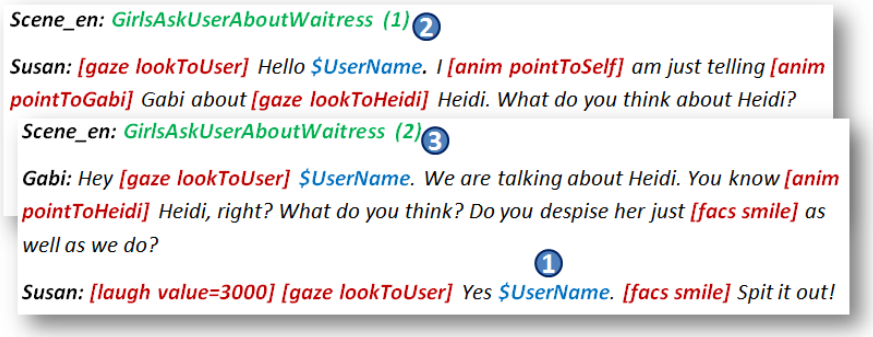

Figure 1: Parameterized scenes of a scenegroup.

A scenescript may provide a number of variations for each scene that are subsumed in a *scenegroup*, consisting of the scenes sharing the same name or signature (Figure 1 (2, 3)). Different *blacklisting strategies* are used to choose one of the scenes from a scenegroup for execution. This mechanism increases dialogue variety and helps to avoid repetitive behavior of virtual characters, which would certainly impact the agents' believability.

1.1.1.2 Modelling Dialogue Content and Logic

A sceneflow is a hierarchical and concurrent state chart that consists of different types of nodes and edges. A scenenode can be linked to one or more scenegroup playback- or system commands and can be annotated with statements and expressions from a simple scripting language, such as type- and variable definitions as well as variable assignments and function calls to predefined functions of the underlying implementation language (Figure 2 (1)). A supernode extends the functionality of scenenodes by creating a hierarchical structure. A supernode may contain scenenodes and supernodes that constitute its subautomata. One of these subnodes has to be declared the startnode of that supernode (Figure 2 (2)). The supernode hierarchy can be used for type and variable scoping. Type definitions and variable definitions are inherited to all subnodes of a supernode. The supernode hierarchy and the variable scoping mechanism imply a hierarchy of local contexts that can be used for context-sensitive reaction to user interactions, external events or the change of environmental conditions.

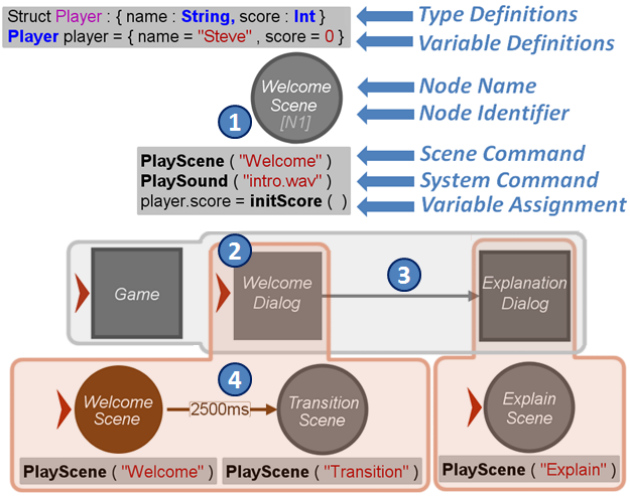

Figure 2: Interruptive conditional edges and simple non-interruptive conditional edges

Different *branching strategies* within the sceneflow, e.g. logical and temporal conditions or randomization, as well as different *interaction policies*, can be modeled by connecting nodes with different types of edges. An *epsilon edge* represents an unconditional transition (Figure 2 (3)). They are used for the specification of the order in which computation steps are performed and scenes are played back. A *timeout edge* represents a timed or scheduled transition and is labeled with a *timeout value* (Figure 2 (4)). Timeout edges are used to regulate the temporal flow of a sceneflow's execution and to schedule the playback of scenes and computation steps. A *probabilistic edge* represents a transition that is taken with a certain probability and is labeled with a *probability value* (Figure 5 (2)). Probabilistic edges are used to create some degree of randomness and desired non-determinism during the execution of a sceneflow. A *conditional edge* represents a conditional transition and is labeled with a *conditional expression*, as shown in Figure 3. Conditional edges are used to create a branching structure in the sceneflow which describes different reactions to changes of environmental conditions, external events or user interactions.

1.1.1.3 Interaction Handling Policies

User interactions as well as other internally or externally triggered events within the application environment can rise at any time during the execution of a model. Some of these events need to be processed as fast as possible to assert certain real-time requirements. There may, for example, be the need to contemporarily interrupt a currently running dialogue during a scene playback in order to give the user the impression of presence or impact. However, there can also be events that may be processed at some later point in time allowing currently executed scenes or commands to be regularly terminated before reacting to the event. These two different interaction paradigms imply two different *interaction handling policies* that find their syntactical realization in two different types of interruptibility or inheritance of conditional edges:

- *Interruptive conditional edges* (Figure 3 (1, 3)) are inherited with an interruptive policy and are used for handling of events and user interactions requiring a fast reaction. Whenever an interruptive conditional edge of a node can be taken, this node and all descendant nodes may not take any other edges or execute any further command. These semantics imply that interruptive edges that are closer to the root have priority over interruptive edges farther from the root.

- *Non-interruptive conditional edges* (Figure 3 (2, 4)) are inherited with a non-interruptive policy, which means that a non-interruptive conditional edge of a certain node or supernode can be taken after the execution of the node's program and after all descendant nodes have terminated. This policy is implicitly giving higher priority to any conditional edge of nodes that are farther from the root.

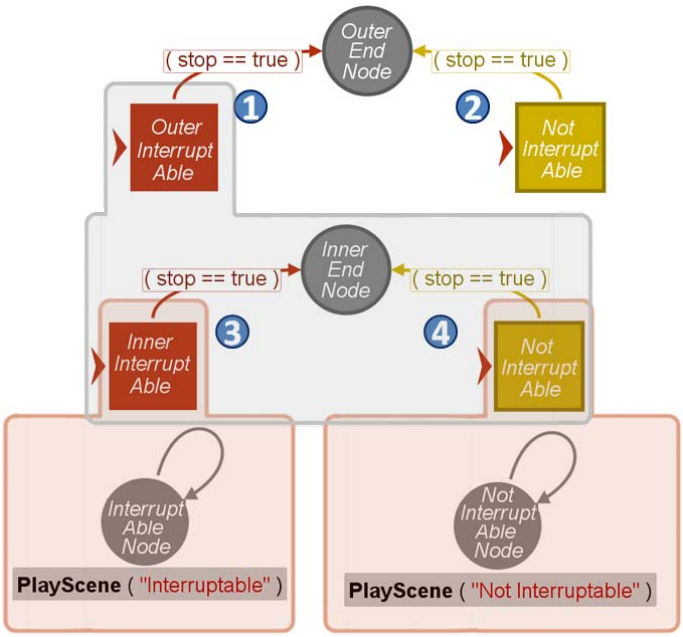

Figure 3: Interruptive conditional edges and simple non-interruptive conditional edges

Figure 3 shows a supernode hierarchy with different conditional edges. If the condition *stop* becomes true during the execution of the two innermost scene playback commands, then the scene within the supernodes with the non-interruptive conditions (Figure 3 (2, 4)) will be executed to its end. However, the scene within the supernodes with the interruptive conditions (Figure 3 (1, 3)) will be interrupted as fast as possible. In the non-interruptive case the execution of the sceneflow continues with the inner end node (Figure 3 (4)) before the outer end node is executed (Figure 3 (2)). In the interruptive case the execution of the sceneflow immediately continues with the outer end node (Figure 3 (1)) because the outer interruptive edge has priority over the inner interruptive edge (Figure 3 (3)).

1.1.1.4 Modelling Multithreaded Dialogue

Sceneflows exploit the modeling principles of *modularity* and *compositionality* in the sense of a *hierarchical* and *parallel decomposition*. Multiple virtual characters and their behavior, as well as multiple control processes for event detection or interaction management, can be modeled as concurrent processes in parallel automata.

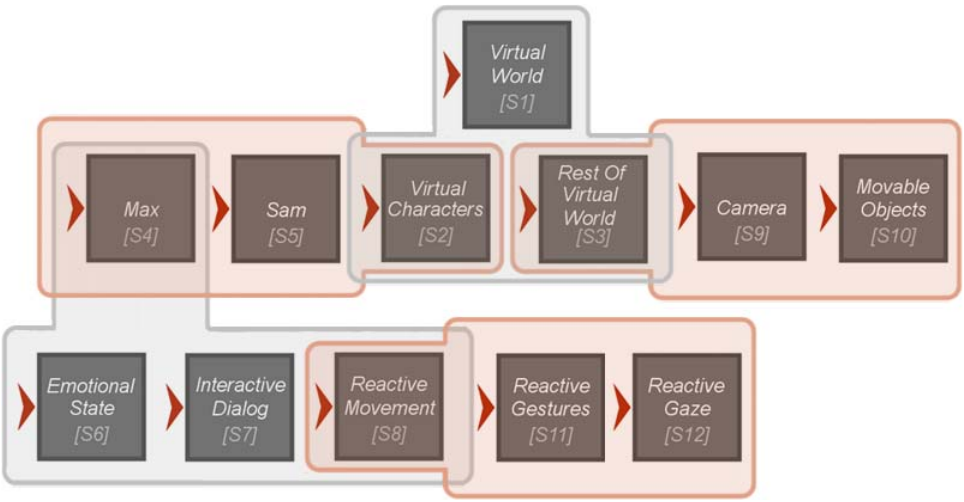

Figure 4: Hierarchical and parallel decomposition.

For this purpose, sceneflows allow two syntactical instruments for the creation of concurrent processes: (1) By defining multiple startnodes for a supernode, as shown in Figure 4, each subautomaton which consists of all nodes reachable by a startnode, is executed by a separate process, (2) by defining *fork edges* (see Figure 5 (1)) an author can create multiple concurrent processes without the need for changing the level of the node hierarchy.

Following this modular approach, an author is able to separate the task of modeling the overall behavior of a virtual character into multiple tasks of modeling individual behavioral aspects, functions and modalities. Behavioral aspects can be modified in isolation without knowing details of the other aspects. In addition, previously modeled behavioral patterns can easily be reused and adopted. Furthermore, pre-modeled automata that are controlling the communication with external devices or interfaces can be added as plug-in modules that are executed in a parallel process.

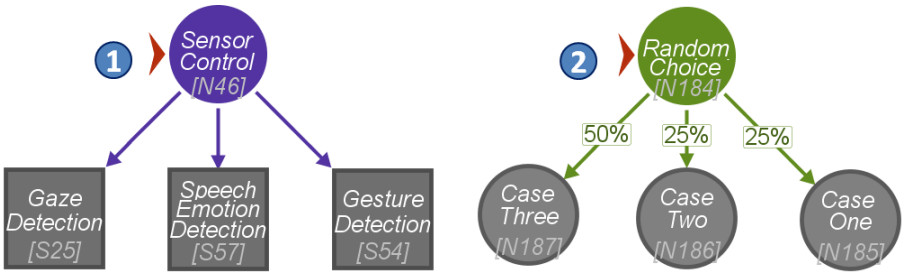

Figure 5: Concurrent processes with fork edges and randomization with multiple probability edges

Individual behavioral functions and modalities that contribute to the behavior of a virtual character are usually not completely independent, but have to be synchronized with each other. For example, speech is usually highly synchronized with non-verbal behavioral modalities such as gestures and body postures. When modeling individual behavioral functions and modalities in separate parallel automata, the processes that concurrently execute these automata have to be synchronized by the author in order to coordinate all behavioral aspects. This communication is realized by a *shared memory model* which allows an asynchronous non-blocking synchronization of concurrent processes.

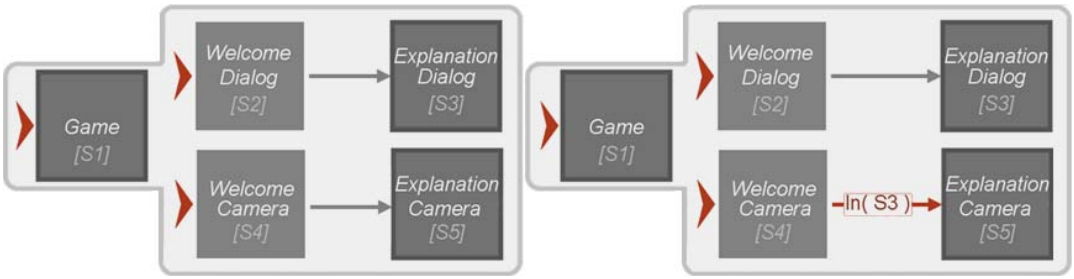

Figure 6: Synchronization over configuration states.

Thereby, sceneflows enfold two different syntactic features for the synchronization of concurrent processes. First, they allow the synchronization over common *shared variables* defined in some supernode. The interleaving semantics of sceneflows prescribe a mutually exclusive access to those variables to avoid inconsistencies. Second, they enfold a *state query condition*, as shown in Figure 6, which represents a more intuitive mechanism for process synchronization. This condition allows requesting whether a certain state is currently executed by the sceneflow interpreter during the execution of a sceneflow.

1.1.1.5 Consistent Resumption of Dialogues

Our concept of an exhaustive *runtime history* facilitates modeling reopening strategies and recapitulation phases of dialogues by falling back on automatically gathered information on past states of an interaction. During the execution of a sceneflow, the system automatically maintains a *history memory* to record the runtimes of nodes, the values of local variables, executed system commands and scenes that were played back. It additionally records the last executed sub states of a supernode at the time of its termination or interruption. The automatically maintenance of this history memory releases the author of the manual collection of such runtime data, thus efficiently reducing the modeling effort while increasing the clarity of the model and providing the author with rich information about previous interactions and states of execution.

The scripting language of sceneflows provides a variety of built-in *history expressions* and conditions to request the information deposited in the history memory or to delete it. The history concept is syntactically represented in form of a special *history node* which is an implicit child node of each supernode. When re-executing a supernode, the supernode starts at the history node instead of its default startnodes. Thus, the history node serves as a starting point for the author to model reopening strategies or recapitulation phases.

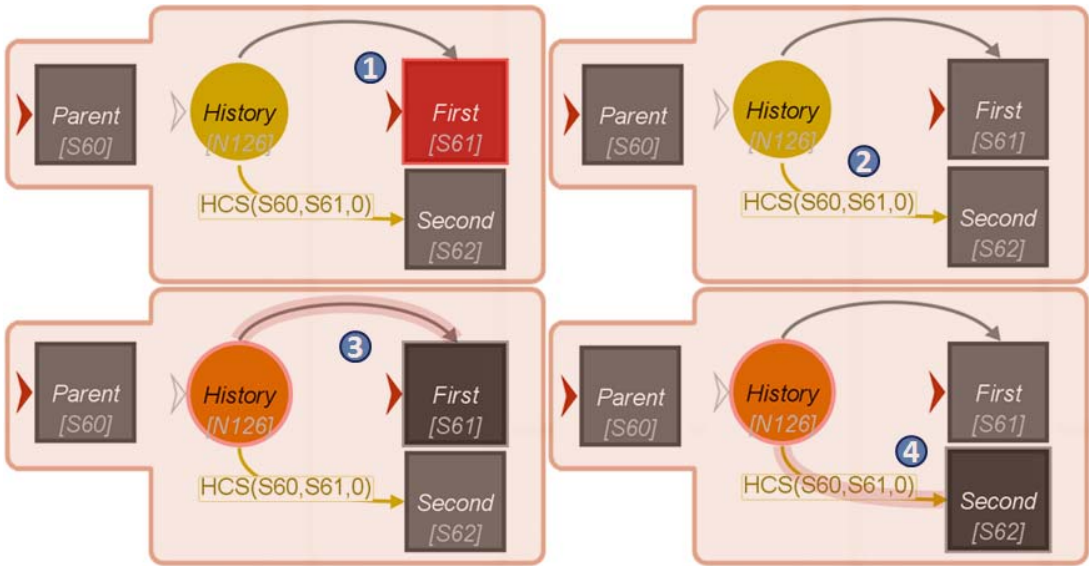

Figure 7: History Node and Condition.

Figure 7 shows a simple exemplary use of a supernode's history node and a history condition. At the first execution of the supernode "Parent", the supernode starts at its startnode "First" (Figure Figure 7 (1)). If the supernode "Parent" is interrupted or terminated at some time and re-executed afterwards, it starts at the history node "History". The history memory is requested (Figure Figure 7 (2)) to find out if the supernode "Parent" had been interrupted or terminated in the node "First" or the node "Second". As the snapshot of the visualized execution shows, depending on the result, the either the node "First" (Figure Figure 7 (3)) is executed or the node "Second" (Figure Figure 7 (4)) is started over the history node.

1.1.2 Creation of Dialogue Domain Knowledge

For an easy and fast creation of the domain knowledge used by the natural language understanding engine we implemented a graphical editor tool. This tool allows a faster creation and modification of the dialogue domain knowledge and a direct testing of the engine's classification quality. Figure 8 and Figure 9 show some screenshots of parts of the editor's graphical user interface. It enfolds a dialogue act editor (Figure 8 (1)) allowing the definition of dialogue acts as they are defined by the ISO standard (Bunt, Schiel, & Steffen, 2004). These formal definitions are based on the latest research results on dialog act annotation and the employed communicative functions directly result from the exemplary dialogue writing in the prototyping process as well as extensions and adaptations from the user testing phase. The definition of some dialogue act ontology can be used in the context of a certain scene or a specific dialogue situation or can be a more general taxonomy of core dialog act concepts. A special ontology or production rule set may be referred to at any point of the execution of the dialogue model created with the Scenemaker modeling tool. This allows adopting or extending the language understanding process to the dialogue context during a scene or a story.

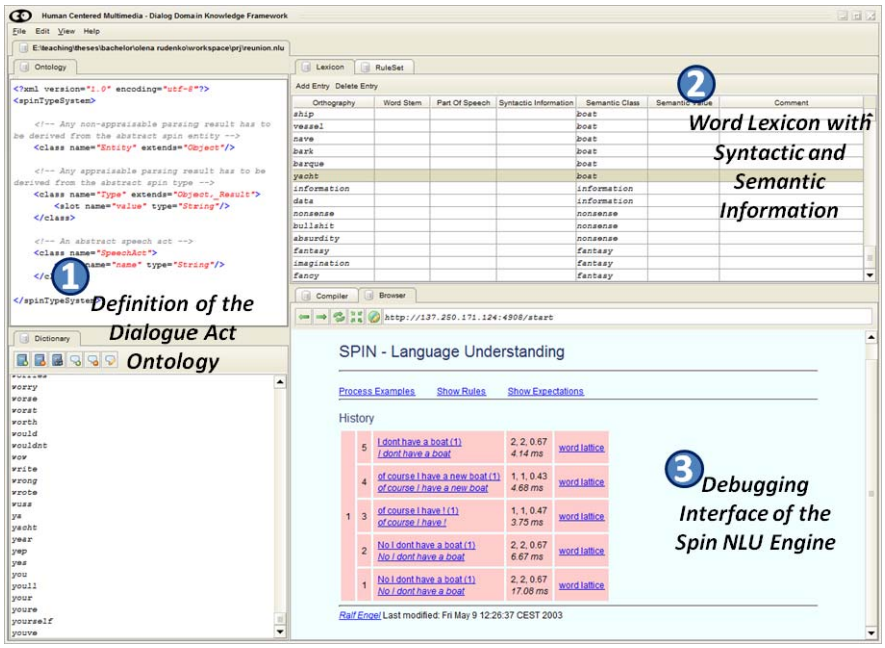

Figure 8: Snapshot of the dialog domain knowledge editor containing a dialogue act ontology editor, a word lexicon table as well as an interface to the SPIN debugger.

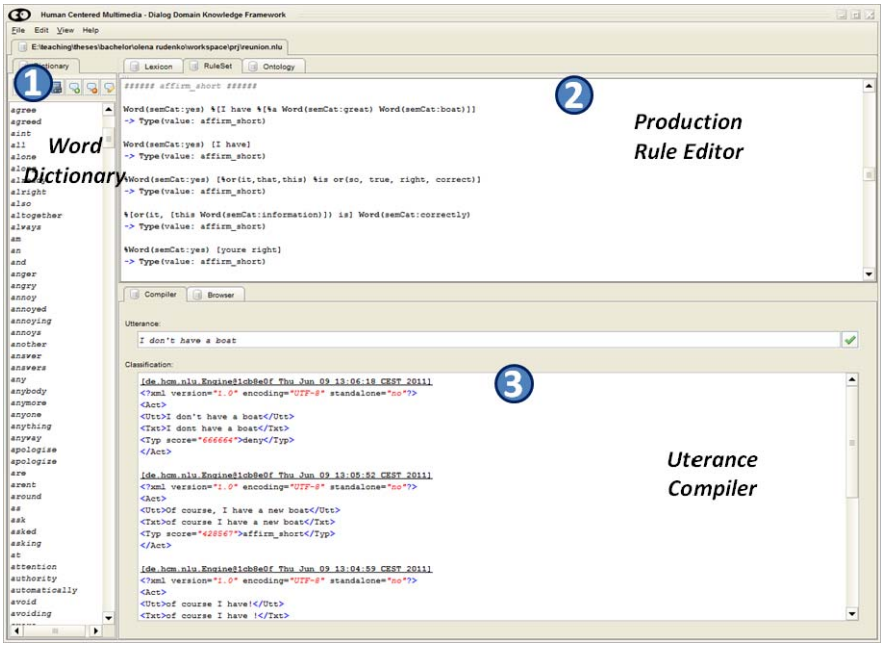

Figure 9: Snapshot of the dialog domain knowledge editor containing a word dictionary, a production rule editor and an utterance compiler for debugging.

As shown in Figure 8 and Figure 9, the editor also allows the creation of the following knowledge base content:

- A *dialogue act ontology* editor (Figure 8 (1)) editor allowing the definition of dialogue acts as they are defined by the ISO standard (Bunt et.al. 2010). As shown in Figure 10, a dialog act contains a communicative function as well as semantic content. The

semantic category is the semantic content, which describes the objects, properties, relations, actions and events that the dialogue act is about. In the case of the semantic parser Spin (Engel, 2005), the semantic information has the form of special keywords that can be extracted from the user's utterances or of more complex structures from an entity taxonomy that are derived from the production rules.

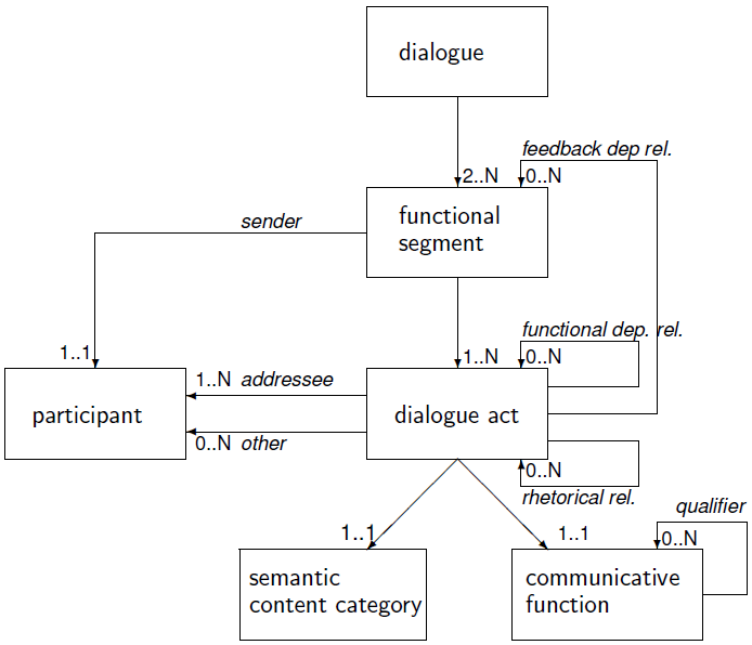

Figure 10: ISO Meta-model for dialogue act annotation.

- A *word lexicon* (Figure 8 (2)) containing syntactic and semantic information for a list of words. The lexicon is used to simplify the production rule set by reducing the amount of necessary production rules.
- A *word dictionary* containing (Figure 9 (1)) all words known by the spell checker engine. The user's utterances are spell checked before they are annotated with dialogue acts. This serves for the handling of typos as well as to restrict the allowed words to be processed to a certain set of words.
- A *production rule editor* (Figure 9 (2)) to specify the parser rules for the translation of the user's concrete utterances into abstract dialogue acts. The editor possesses a syntax highlighting mechanism and falls back to the SPIN engine's built-in syntax checker to detect grammatical, lexica l or syntactical errors in the specification of the production rule set.
- An *utterance compiler* (Figure 9 (3)) allowing the direct testing and debugging of the modifications and adoptions of the knowledge base content. It enfolds a text field for the user input and an output area which displays the parsing results in a simple xml format. The resulting xml sections contain the user's typed utterance, the corrected and spell checked utterance, the abstract dialog act with its communicative function as well as a possibly empty list of extracted semantic content in form of keywords.

1.1.3 Integration of Pre-Authored Story Elements

Since we observed several shortcomings of the 3D game engine with respect to the technical feasibility of the story content provided by the external storywriter, we thought about several alternative representation layers that could be used to augment or replace the current *Virtual Beergarden* application. The different representation layers rely on different types of media and can be used to display a variety of pre-authored story elements. The different representation layers can be used in parallel with each other and the conventional 3D game engine representation and can be arbitrarily intermixed.

1.1.3.1 Integration of Different Representation Layers

The Scenemaker runtime environment allows the integration of runtime plug-in modules for the generation of output, the interpretation of input or both at the same time. We implemented several plug-in modules for the Scenemaker modeling tool in order to be able to experiment with different media representations and to evaluate the effects of these representations on the storytelling experience in the future. Figure 11 shows an overview of the software components that are part of the core Scenemaker runtime environment and the components that have been implemented as plug-in modules as well as the knowledge bases and data repositories, for example the dictionary for the semantic parser (Figure 11 (8)), used by these components. A simple Scenemaker project consists of at least one sceneflow (Figure 11 (1)) that has a reference to at least one scenescript (Figure 11 (2)) containing the pre-scripted dialogue content. Scenes in a scenescript can contain action tags that refer to different knowledge bases and data repositories. The scenes are executed by the sceneplayer (Figure 11 (4)) that is called at runtime from within the sceneflow interpreter (Figure 11 (3)). These tags can refer to animations in an animation lexicon, as in the previous version of the SOAP storytelling application, however, these tags may also contain commands that refer to data files such as video, audio or image files that should be displayed or played back during the execution of a scene. So we considered the following additional representation layers and implemented them as plug-in modules for the Scenemaker modeling tool:

- An *audio player* plug-in (Figure 11 (6)) can be used to play back recordings of human actors' voices instead of using the usually used TTS engine.
- A *video player* plug-in (Figure 11 (5)) can be used to play back recordings of videos of human actors performing scenes from the scenescripts instead of using the 3D game engine.
- A *text-based* user interface (Figure 11 (7)) can be used as a tool for the early testing of the modeled dialogue.

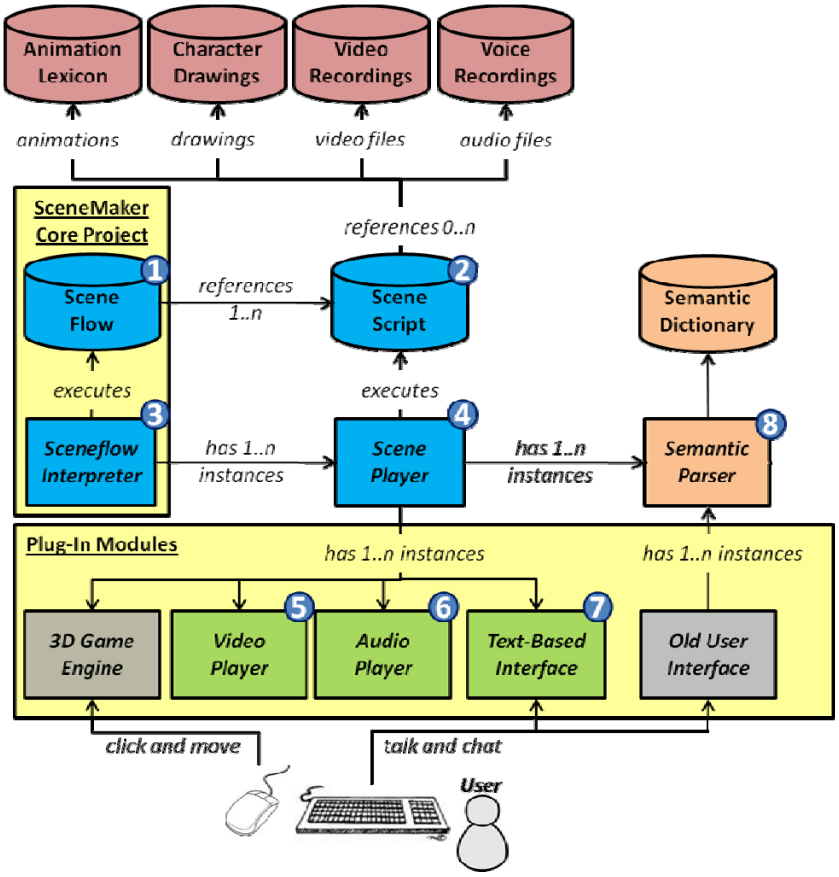

Figure 11: The System Component Structure.

1.1.3.2 Integration of an Audio Player Plug-In

We considered the recording and playback of human actor‘s videos instead of using the TTS engine. This would certainly improve the agents’ believability because voice recordings allow a much broader range of emotional vigor, intonation, accent and other properties of human voice. For that reason we extended the Scenemaker tool with a plug-in that allows to playback video files.

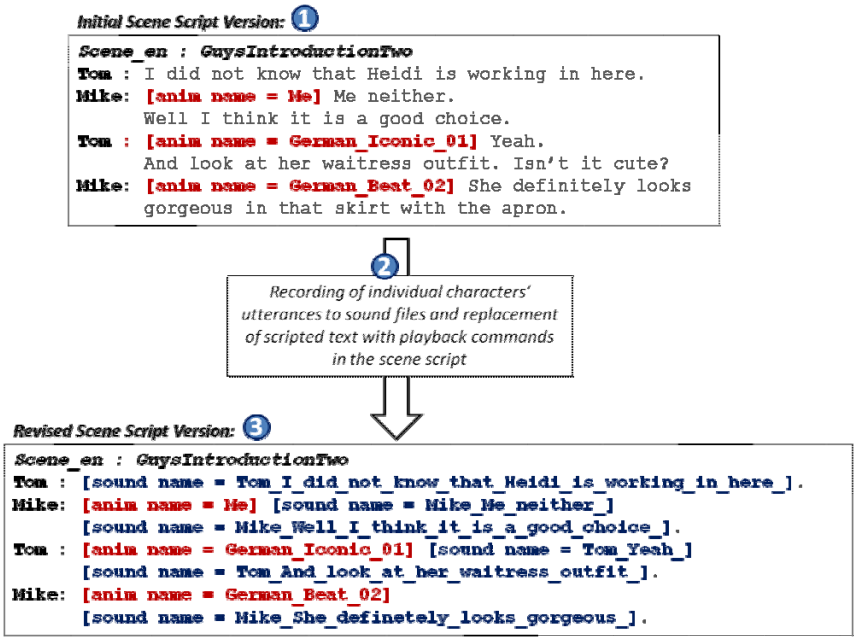

Figure 12: The integration of sound playback into a scene

As shown in Figure 12, the sound files that have to be played back can easily be specified within the conventional scene script format. At runtime the Sceneplayer instance resolves the file name, loads the sound file and delegates the playback of the sound file to the sound player plug-in. The playback of recorded sounds can arbitrarily be intermixed with animations played back in the Virtual Beergarden application and with TTS commands or videos or can be played back in parallel.

1.1.3.3 Integration of a Video Player Plug-In

We also considered the recording and playback of human actor‘s videos instead of using the TTS engine. This would certainly improve the agents, or actors' respectively, believability because video recordings allow a much broader range of acting performances. For that reason we extended the Scenemaker tool with a plug-in that allows to playback video files.

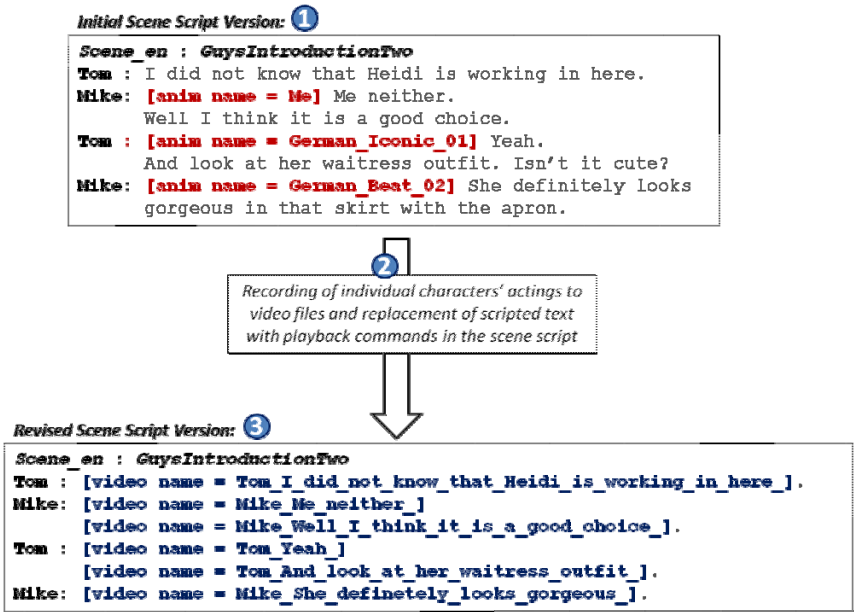

Figure 13: The integration of video playback into a scene.

As shown in Figure 13, the video files that have to be played back can easily be specified within the conventional scene script format. At runtime the Sceneplayer instance resolves the file name, loads the video file and delegates the playback of the video file to the video player plug-in. The playback of recorded videos can arbitrarily be intermixed with performances played back in the Virtual Beergarden application and with TTS commands or sounds or can be played back in parallel.

1.1.3.4 Integration of a Text-Based User Interface

A *text-based* user interface (see Figure 14) can be used as a tool for the early testing of the modeled dialogue and the domain knowledge, especially the adequate reaction to the users discourse acts. The dialogue progress is logged into a log file for later analysis.

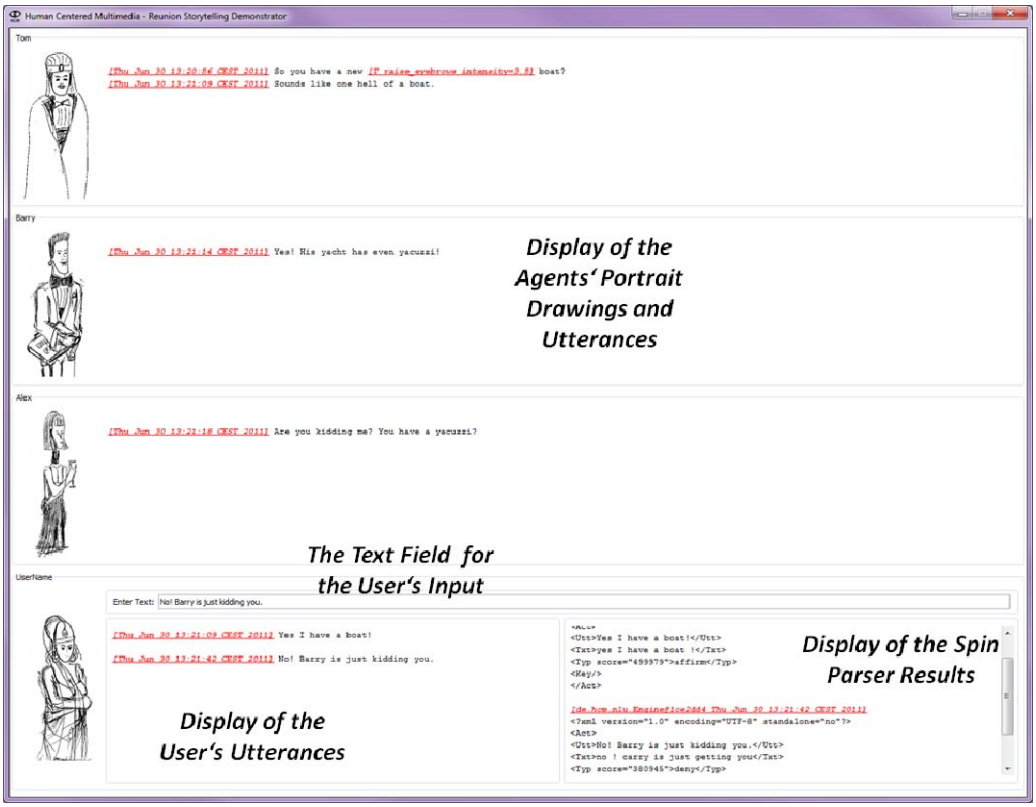

Figure 14: The Text-Based Interface.

1.1.4 Testing of the Dialogue Managament Tools with Design Students and a Professional Script Writer

We recently conducted a hands-on workshop on the modeling tool *Scenemaker* and the *Virtual Beergarden* application together with students of a media management course at the HSRM. Aim of this two day lasting workshop was to introduce the students into the handling of *Scenemaker*'s graphical user interface and the modeling principles with the state chart variant used by the modeling tool.

In the course of the student project we will improve and finish the implementation of our graphical editor for the creation of dialogue domain knowledge used as modeling interface to the semantic parser *Spin*. We will present this framework to the students and give them an introduction into the specification of dialogue act ontologies and the production rule language used by the semantic parser. With the help of this tool they will hopefully be able to create their own dialog domain knowledge and parsing rules for the dialogues they shall realize for the Reunion scenario written by the external author.

The story written by the external storywriter Georg Struck shall be realized, at least in parts, in the future by the students of the student project at the HSRM and the participants of the summer school. For that reason we expect drafts and prototypical descriptions of the scenes and dialogue situations within the story by Georg Struck at regular intervals. Individual scenes and dialogues shall be specified and submitted in a semiformal format by Georg Struck and subsequently be realized and tested with our tool framework.

The planned cooperation with the HSRM and the external storywriter in the future can be outlined as follows. We will conduct a hands-on workshop on the modeling tool *Scenemaker* and the *Virtual Beergarden* application together in the context of the interactive storytelling

summer school held at the HSRM.

Recently we received a first draft of the story containing the semiformal specification of a first single exemplary dialogue situation. As shown in 15, the draft also contains the first partial ordered set of scenes in the story and a short description of the story content of each scene.

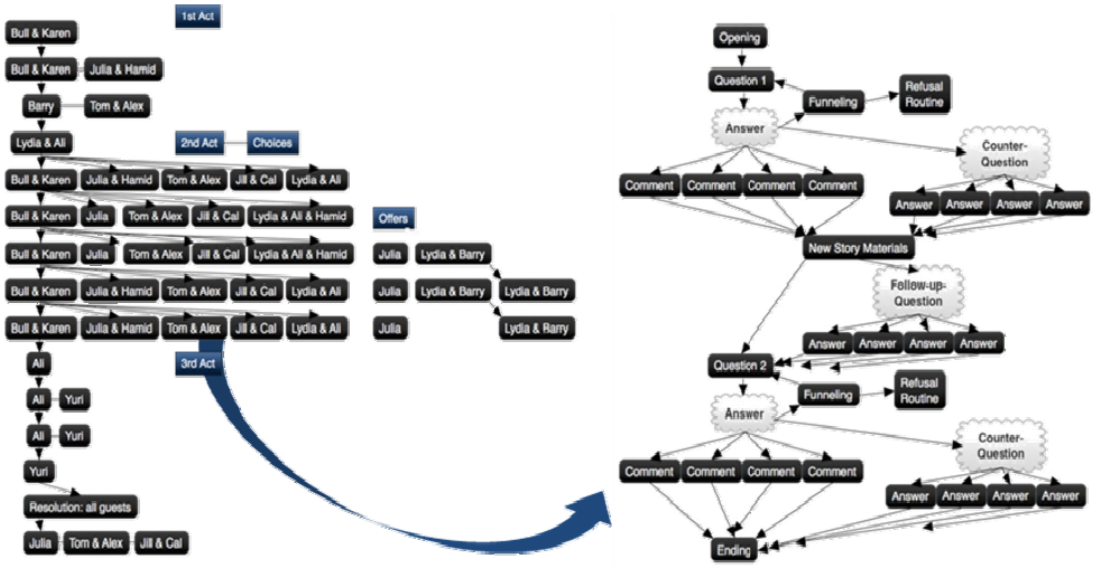

Figure 15: Iterative authoring process in several prototyping steps.

This descriptive state based specification will then be implemented with our tool framework into the formal models, namely the state chart variant used by Scenemaker as well as the adequate dialog act ontologies and production systems for the natural language understanding engine.

1.2 FULL BODY INTERACTION IN INTERACTIVE STORY TELLING

Our current research also includes full body interaction enabled by user skeleton tracking on a depth sensor. With the release of the Kinect sensor (http://www.xbox.com/kinect) in November 2010, Microsoft has made depth sensors easily available for every home. While the Kinect is originally an additional peripheral for the Microsoft Xbox 360 gaming console, it can also be connected to a PC. This enables to take the next step in human computer interaction by controlling your PC with gestures and movements. Although, it is a challenging task to apply this new type of interaction and to find possibilities for offering a good user experience and usability along with it.

We already have investigated several ways of interacting with full body tracking and therefore implemented different gesture recognizers, avatar controls, and interaction techniques with graphical user interfaces.

1.2.1 Kinect Hardware, Drivers, and Tracking

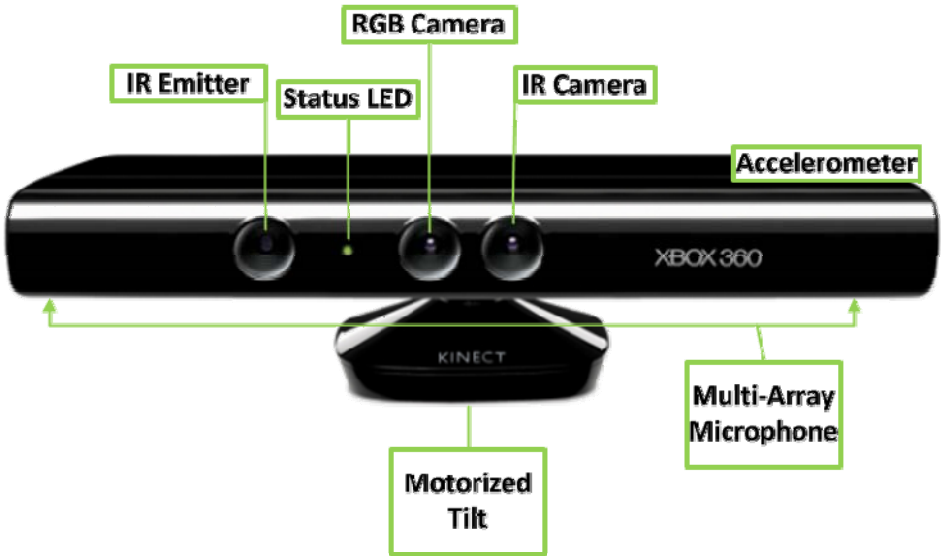

Figure 16: Kinect hardware components

Figure 16 is an overview of the Kinect hardware components that have the following functionality:

- The status LED changes its color according to Kinect's connection status.
- The accelerometer data gives information about how the Kinect is positioned, while its main purpose with the Xbox 360 is to detect whether the Kinect was moved. This pauses any current game and asks for recalibration.
- The tilt motor is used to adjust the viewing angle of the sensor.
- The multi-array microphone uses ambient noise suppression and echo cancellation, can recognize spoken keywords via the Microsoft Speech API and additionally localizes the acoustic source.
- The RGB camera acts like a normal web cam.
- IR emitter and camera provide a depth image created according the structured light principle:
    - IR emitter projects the scene with an irregular pattern of IR dots
    - IR camera captures the dots on the environment (see Figure 17)
- A depth image is reconstructed by recognizing the transverse shift of the dots in relation to a reference image at a known distance (PrimeSense Ltd. Patent No. US2010/0118123)

Figure 17: IR image the Kinect uses for calculating a depth image

We are currently using OpenNI (Open Natural Interaction) for utilizing the Kinect on the PC. The OpenNI Framework is an open source framework for natural interaction devices (especially depth sensors). It offers an interface for accessing the data of those devices, passing the data to other middleware modules and getting back the results of those modules.

The framework is developed and distributed by the OpenNI organization, an industry-led, not-for-profit organization (http://openni.org). Its current members are PrimeSense (http://www.primesense.com), Willow Garage (http://www.willowgarage.com/), Side-kick (http://www.sidekick.co.il/) and ASUS (http://www.asus.com/).

While ASUS additionally distributes its X-Tion Pro – an OpenNI compliant depth sensor, PrimeSense offers its OpenNI compliant middleware NITE (Natural Interaction). NITE provides user skeleton tracking with several important joints on the data stream of a depth sensor. Figure 18 visualizes the user tracking on the depth stream and the simplified skeleton fitted on the 3D shape of the user. The tracking is relatively stable, but it has some known issues with near walls, occlusion, or hands near the body (Figure 18, right-hand side). NITE practically needs the user to stand in a specific calibration pose before tracking, but this calibration can be saved and reapplied later. It also can be applied to a different user as a default calibration, that may not be as accurate as an own calibration, but works well in practice.

The depth image calculation of the ASUS X-Tion Pro is the same as for the Microsoft Kinect, since both include the PrimeSense PS1080 System on Chip for that task.

It is therefore no surprise, that there exists an inofficial OpenNI compliant driver for the Kinect (https://github.com/avin2/SensorKinect) that enables to use the Kinect with OpenNI and the NITE tracking. It also gives access to the raw RGB, IR and depth image.

Nevertheless, Microsoft has released their Microsoft Kinect SDK beta in June 2011 (http://research.microsoft.com/kinectsdk), that includes an own driver and user skeleton tracking. It additionally gives access to Kinect's microphone array.

With a special OpenNI module (https://www.assembla.com/code/kinect-mssdk-openni-bridge/git/nodes/), the driver and tracking of the Microsoft Kinect SDK beta can also be used with OpenNI.

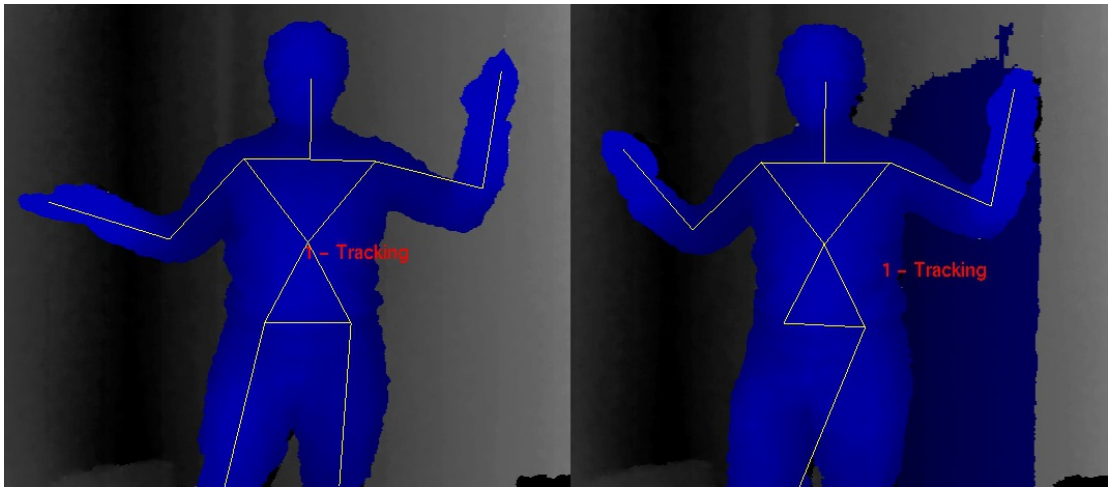

Figure 18: User skeleton tracking with NITE

The tracking of the Microsoft SDK beta does not need any calibration pose, as it uses a significantly different tracking algorithm that works on a per-frame basis (Shotton, et al., 2011). It also has fewer problems with near walls, arms near the body, and occlusions, but these problems are still not solved completely. Furthermore, the Microsoft SDK beta currently supports only two users to be tracked in parallel and gives no access to the IR image.

1.2.2 Gestures and Postures with Full Body Tracking

We distinguish four different gesture and posture categories:

1. *Static postures:* They include no movement, but describe specific relations between the tracked joints, and consequently the configuration of a part of the skeleton. They can be directly read from the current joint-coordinates in every frame. Examples are “lean forward”, “left arm up”, and “hands above the head”.
2. *Gestures with linear movement:* They describe a linear movement or velocity of one or more joints. They need the calculation of the movement between two or more frames depending on smoothing. Examples are “left hand moves right” and “hands move up”.
3. *Combination of postures and linear movement:* They are realized as switching between a set of static postures and linear movements of 1 and 2 following specific time constraints. In that manner, we can for example recognize waving as “hand over shoulder, alternating moving left and right” or walking without moving with “knees alternating moving up and down”.
4. *Complex gestures:* This includes a more detailed observation of one or more joints (mostly the hands or the shoulder to hand vector) over a certain amount of time (and depth frames). On the one hand, that can be used directly to simulate an input device. For example a mouse-like control can be implemented with one hand joint, using the x,y-movement for the cursor movement on screen and the z-movement (e.g. “push gesture”) or hovering over an object of the graphical user interface for clicking. Multitouch input could be implemented by looking at both hands, or the fingers (but finger tracking is currently not supported by the software). On the other hand, the detailed observation can be done for gesture recognition with a specific algorithm, such as the Dollar$1 algorithm (or Protractor3D (Li, 2010), an improved 3D version of the Dollar$1).

Besides this gestures and postures, further information about the complete body in the scene

can be used to recognize movement and orientation in the scene or jumping.

1.2.3 Toolkit for Posture and Gesture Recognition with Kinect

We have implemented a toolkit for posture and gesture recognition with the Kinect.

The most important features are:

- Easy access to the tracking data of OpenNI/NITE or OpenNI/Microsoft Kinect SDK
- Debug image with tracking info for each user
- Example posture and linear movement recognizers and an interface for creating your own
- Example combined posture/linear movement recognizers and an interface for creating your own
- Organization of users:
    - Default calibration and calibration for specific users (only needed for NITE)
    - Saving pictures of the user's face, e. g. for later face recognition
- Recognition of complex gestures with SSI (Wagner, Lingenfelser, & André, 2011) and the Dollar$1 algorithm (Wobbrock, Wilson, & Li, 2007)

With this toolkit, we already realized a graphical user interface (GUI) with Kinect support for our Horde3D GameEngine (http://www.hcm-lab.de/projects/GameEngine). As described in Section 1.2.2, we are using the hand movements for controlling a cursor on screen (left or right hand or two cursors with both hands if needed). We also implemented several actions for GUI events, i.e. hover, click by hover duration, click by push gesture, click by wiping gesture.

Figure 19 depicts one application with the Kinect GUI. The user controls the cursor (orange hand) on screen with his right hand and can write text in the white text field by "clicking" letters on the virtual keyboard. This is done by hovering over a letter with the cursor for 1.5 seconds or by performing a "push gesture" while hovering. In the lower right corner is the provided debug image that shows the tracked shape of the user (cyan figure).

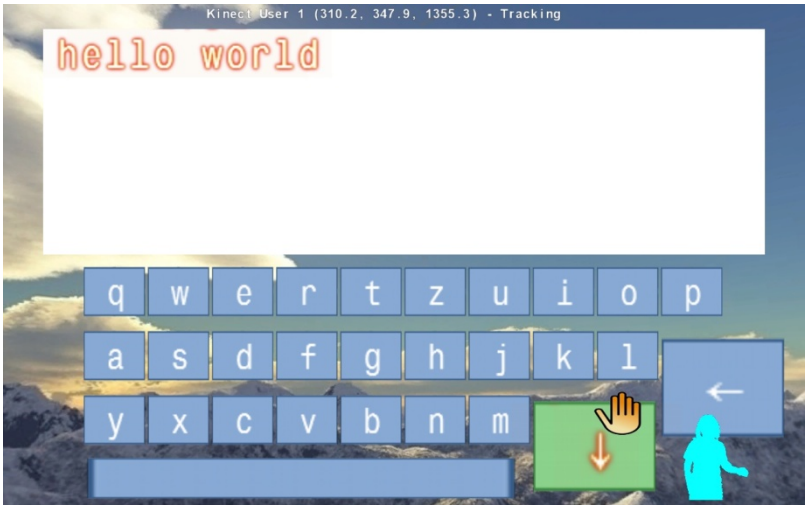

Figure 19: GUI with Kinect support

2. Evaluation of Interaction

In this section, we summarize the results of two studies we conducted to evaluate different styles of interaction focusing on gesture-based and language-based interaction. Unlike the experiments we presented in Deliverable D6.3, we made use of the newly developed measurement toolbox developed within IRIS for IS systems.

2.1 Evaluation of Full Body Interaction in Interactive Story Telling

2.1.1 Background of Research

Various approaches for providing innovative interaction modalities in interactive storytelling scenarios have been investigated in the past, some of them are described representatively in the following, though none of them uses full body interaction.

(Cavazza, Charles, & Mead, 2002) presented an interactive storytelling system where the user can influence the story by talking to a virtual character and giving advice, such as 'don't be rude'. To this end, they integrated a speech recognition component that makes use of template matching.

Façade is a well-known interactive drama by (Mateas & Stern, 2003). The user can interact by typing any kind of text message using the keyboard. Façade was further developed to an augmented reality version, additionally implementing speech recognition within a Wizard-of-Oz setting to enable more natural interaction (Dow, Mehta, Harmon, MacIntyre, & Mateas, 2007).

The Geist project is an augmented reality storytelling system (Braun, 2003) with which the users can interact by performing gestures on a magic wand and uttering simple sentences.

We have implemented a prototypical storytelling application with the Toolkit described in Section 1.2.3. To this end, we adopted the non-linear story of the game book “Sugarcane Island” by Edward Packard (Packard, 1986). We added so-called Quick Time Events (QTEs) that are a frequently used instrument in recent video games. Whenever a QTE occurs in a video game, a symbol representing a specific action on the control device appears on screen. The user then has a limited amount of time to perform that action in order to complete the QTE successfully. In most cases, the successful performance of the QTE leads to a special action of the player avatar while an unsuccessful performance causes the player avatar to fail with this action. The utilization of QTEs ranges from enriching cut-scenes with interactivity to using them as the main gameplay mechanic.

To our knowledge, there have been no scientific studies on the application of QTEs yet. However, a good example of the industry, that excessively uses QTEs are the games 'Fahrenheit' ('Indigo Prophecy') and 'Heavy Rain' by the game developer Quantic Dream. For implementing the QTEs, they use mouse gestures and keyboard input on the PC. On the game console, they use button and analog stick input as well as gestures performed with a game pad including an accelerometer. In this way, the user needs to perform abstract actions or gestures on a controller in opposite to our implementation of QTEs.

2.1.2 Full Body Interaction in an IS Application

We adopted the idea of QTEs for full body interaction with Kinect. As soon as a QTE starts in our application, a countdown appears centred at the top of the screen and for each user one symbol is shown, representing the requested action as shown in Figure 20. As soon as one user solves a QTE, a mark is shown on top of the symbol providing immediate feedback. The

whole QTE is solved, if both users successfully completed their action before the countdown reached zero.

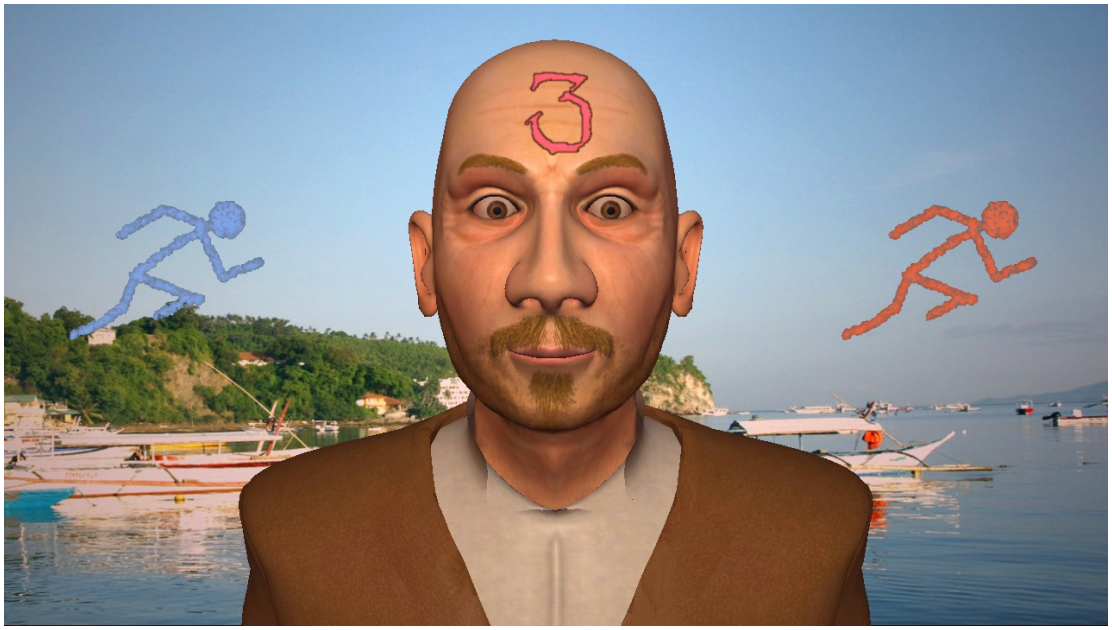

Figure 20: Screenshot of our application during a QTE

We implemented two different modes for the realization of QTEs, both using the Kinect with our Toolkit:

In the first mode, each user has to press a randomly positioned and sized button on screen with a cursor controlled by moving the hand.

In the other mode, the users need to perform gestures that are indicated on screen by one of the symbols presented in Figure 21.

Figure 21: Gesture symbols

The requested user actions for each symbol are (from left to right):

- *Balance:* Move the hands out in a height equal to the shoulder.
- *Kick:* Perform a kick with one leg.
- *Catch:* Put the hands together in front of the body.
- *Climb:* Move the hands up and down in front of the head, like climbing.
- *Left and right Hand:* Raise the left or right hand.
- *Run:* Move the knees up and down in place, like running without moving.

Figure 22: Two users performing a QTE gesture during our application

Figure 22 shows two users in front of a screen, performing the QTE gestures “Kick” (left user) and “Catch” (right user). Note the Kinect placed centred below the screen. The left user has already succeeded in the QTE “Kick”, so a green Check mark has appeared over the corresponding symbol.

2.1.3 Results of User Study

To evaluate the two different modes for the realization of QTEs, we conducted an experiment with 18 participants. The participants were arranged in pairs. We applied a within participants design and thus each pair had to participate in both conditions (i.e. button and gesture mode). To prevent ordering effects we counterbalanced the order of the conditions. The average age of the participants was 24.7 years and we applied two-tailed paired t-tests to show the validity of the results.

The recognition within the button and the gesture mode turned out to be very robust. The participants succeeded in 93% of all actions within the button-based QTEs (i.e. 67 out of 72 possible quick time button actions). For the gesture-based QTEs, the participants were even more successful with 97% of all possible actions (i.e. 65 out of 67).

To evaluate the users' subjective impression, we made use of a questionnaire inspired by the IRIS Evaluation and Measurement Toolkit. Each statement was given on a nine-point Likert scale ranging from 'strongly disagree' (-4) over 'neutral' (0) to 'strongly agree' (4).

We observed that the participants found the gesture based time QTEs (M: 2.9, SD: 0.9) significantly easier to use than the button based events (M: 2.0, SD: 1.9) where they had to simply point at a specific button label ($t(17) = 2:1$; $p < 0:05$; $r = 0:29$). The participants also thought that most people were able to learn the system with the gesture-based QTEs (M: 3.3, SD: 0.8) faster than the one without (M: 2.4, SD: 1.5; $t(17) = 2:2$; $p < 0:05$; $r = 0:35$). The gesture mode (M: -3.1, SD: 1.0) was considered as more comfortable to use compared to the button based mode (M: -1.3, SD: 2.1; $t(17) = 3:5$; $p < 0:01$; $r = 0:48$).

The participants were significantly more satisfied with the gesture based mode (M: 2.3, SD: 1.0) than with the button based one (M: 1.6, SD: 1.4; $t(17) = 2:4$; $p < 0:05$; $r = 0:28$). The gesture mode (M: 0.0, SD: 2.5) was also considered as significantly less inconvenient compared to the button based mode (M: -2.3, SD: 1.7; $t(17) = 4:0$; $p < 0:001$; $r = 0:47$). The interaction with the gesture based mode (M: 3.1, SD: 1.0) was significantly more fun than the button based mode (M: 1.0, SD: 2.3; $t(17) = 3:8$; $p < 0:01$; $r = 0:51$) and finally the participants stared with significantly higher expectations at the screen in the gesture based mode (M: 1.4, SD: 1.6) compared to the button based mode (M: 0.8, SD: 1.5; $t(17) = 2:2$; $p < 0:05$; $r = 0:19$).

The results imply that natural gestures for QTEs do not only contribute to better usability, but are also more comfortable and more enjoyable. We conclude that it can be worthwhile to design new and more natural ways of interaction for a full body tracking system instead of simply adapting the classical point and click paradigm.

2.2 Evaluation of Dialogue-Based Interaction in Interactive Story Telling

2.2.1 Background of Research

The question of how much user initiative is appropriate in dialogue has been mainly addressed in the area of natural language processing so far. Studies conducted by (Walker, Fromer, Di Fabbrizio, Mestel, & Hindle, 1998) for a natural language dialogue system seem to indicate that users prefer a system-initiative interface over a mixed initiative interface. However, robust mixed initiative interfaces are hard to realize technically and the lower user satisfaction with the mixed initiative interface might also been caused by the worse performance of the speech recognizer in this case.

Studies on the appropriate level of user initiative in interactive story telling are still rare. An exception is the work by (Sali, et al., 2010) who investigated three different dialogue approaches for game and interactive story telling interfaces. They found that users prefer a natural language interface over interfaces that allow users to select sentences and interfaces that make use of an abstract response menu interface. However, some users also had problems with the natural language interface because they found it hard to figure out what to say in a particular situation.

(Mehta, Dow, Mateas, & MacIntyre, 2007) evaluated design strategies in the interactive drama Façade by analyzing the user's experience during conversation breakdowns. Their results indicate that user engagement may be maintained in such situations by providing users with sufficient narrative cues to integrate the virtual characters' reactions into the story. Even during conversational breakdowns a believable character performance may still provide an enjoyable experience.

The same system was evaluated by (Roth, Klimmt, Vermeulen, & Vorderer, 2011) in comparison to the adventure game Fahrenheit, while user experience was investigated either after interaction or after watching a video of the application. Their results indicate that for

interactive storytelling systems, such as Façade, the user's experience is strongly related to interactivity.

2.2.2 Dialogue-Based Interaction in an IS Application

As discussed above, the question of how much user initiative is appropriate in dialogue-based interactive storytelling is still not sufficiently explored. Therefore, our objective is to investigate in how far the user's freedom of choice or the guidance by the system influences the user's perception of the experience. This is a crucial question in interactive storytelling systems, since the user's autonomy is proven to be an important feature that influences the user's experience on the entertainment (Ryan, Rigby, & Przybylski, 2006). However it is not clear to what extent the design of a system needs to be adapted in order to reach a feeling of autonomy.

In order to investigate this question, we made use of the Virtual Beergarden application (http://www.informatik.uni-augsburg.de/lehrstuehle/hcm/projects/aaa/) where system-controlled characters as well as a user avatar can interact in the scenario. The characters interact with each other using verbal and nonverbal behaviour and the user who provides typed text as input.

The architecture of our dialog-based interaction scenario makes uses of the following interaction components of the technical framework described in Section 1 (see Figure 23):

- SceneMaker tool for modelling and realizing multi-threaded dialogues between the characters and the user
- Semantic parser Spin for natural language understanding

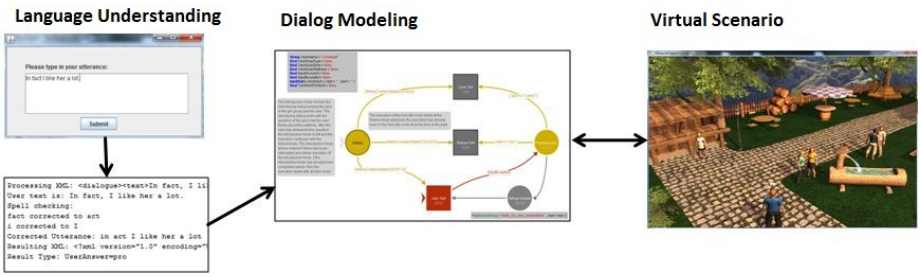

Figure 23: System architecture of the Dialog-based Interaction Scenario.

2.2.3 Interaction Styles

To investigate which level of user initiative is preferred by the user, we developed two different versions of a dialog-based interactive storytelling prototype: One version offering a continuous dialog interaction to users, whereas the other version offers round-based, system-initiated dialog possibilities.

Continuous Interaction

In this version of our demonstrator, the user is able to interact at anytime. With it, we aim on providing interpretational freedom to the users and to enhance their perception of the story by inspiring their curiosity and encouraging their spirit of exploration. Therefore, the graphical user interface for the user's typed text input is always enabled. Thus, the user can interact whenever he or she likes. The user avatar speaks out the typed utterance, while the system-

controlled characters of the target focus group interrupt their current dialog at utterance level and react to the user's input promptly.

This version of our interactive demonstrator is a lot more complex, since agent dialogs need to be interruptive in a manner that characters are able to react to user interaction at any time. Therefore also a much larger set of interaction possibilities needs to be provided as users might interact differently during an ongoing dialog than they might interact after being asked a specific question. With providing a continuous interaction style, user interaction should be a lot more natural and intuitive, since dialogs in real life work in that manner, too. An important issue arising due to this interaction style is the timing of user input and system output.

The dialog management tool of our system provides concepts for hierarchy, concurrency, variable scoping, multiple interaction policies and a runtime history, which is needed for this kind of interaction. As mentioned above, in the continuous version of the system user input might happen at any time. Thus the current dialog needs to be interruptive as a prompt reaction to the user's input is required. Therefore, so called interruptive conditional edges are used. In case of a user input, the dialog that was currently active is stopped at utterance level and all descendant nodes may not take any other edges, or execute any further commands.

In case of user input, the process modeling of the current focus group needs to be properly synchronized with the user agent. The synchronization is realized by reacting promptly to the user's input during a conversation. After a user's utterance has been detected, the conversation either continues with the transition to another dialog or with reentering the previous dialog. For the latter, a runtime history provides the possibility to remember the last sub states of a conversational flow. When, for example a dialog should be reentered that was active before, instead of its default start node, the dialog can be executed starting at the history node.

With this version of the demonstrator, we provide a highly interactive system that might help to engage the user. Potential issues with this version might be a confusion of the user, in case he or she does not know how or at what point in time to interact with the system-controlled characters, since interaction is possible at any time in principle. In addition, the impact of the user's interaction to the flow of the story might not always be clear, since virtual characters do sometimes need to block or redirect the user in order to continue with the story line.

System-Initiated Interaction

For the system-initiated interaction style of our demonstrator, the scene flows and scripts from the continuous version were reused. However, not all states are likely to be reached, since the graphical user interface for the user's typed text input is disabled during agent conversations. Only at points in time when the user can state an opinion or advice a character, and thus make a contribution to the narrative flow, the user is actually able to communicate with the system. Thus, clear interaction prompts are provided by the characters that are modeled in a way that they are asking the user for suggestions at certain points in the story. Thus, only user interactions that are typical for the particular branch are likely to occur. In that manner, on the one hand the user knows what kind of input is required. On the other hand, the parsing of these sentences is easier, since the domain is limited. If the user is, for example, explicitly asked for his or her opinion on the waitress, it is likely that the user states an opinion and does not try to comment on other things. Only a few of the possible characters responses are therefore needed in this version of the demonstrator. In sum, the story is kept the same as for the continuous version but with a lot less possibilities.

Dialogs that are held by system-controlled characters are not interruptive in this version. Only if the end of a dialog is reached, a contribution to the dialog from the user's side is possible. Thus, interruptive edges are not needed. However, history nodes are maintained, since a dialog still needs to be able to be reentered, e.g. if the user walks away and approaches the target group again afterwards.

This interaction style is a lot simpler to model than the continuous version, for the reasons described above. As a possible advantage, it might be more intuitive to use for inexperienced users, since clear questions are asked at certain points in time only. Since interaction is only possible at story branches, the impact of the user's interaction to the flow of the story should be comprehensible to the user. However, this version might be a bit boring for the user.

2.2.4 Results of User Study

An experimental user study was conducted in cooperation with our IRIS project partners from VUA to explore whether the substantial difference that deciding for continuous or round-based user dialogue would make from a designer's point of view is reflected in the end-user experience as well. The research question was therefore whether the user experience would be different in an interactive storytelling prototype that involves continuous dialogue compared to a prototype that involves round-based dialogue.

Conceptually, the term "user experience" was grounded on previous theoretical and empirical work conducted by VUA, see (Klimmt, Roth, Vermeulen, Vorderer, & Roth, in press) and (Vermeulen, Roth, Vorderer, & Klimmt, 2010). Based on theory and research in media psychology, a dimensional framework of possible user responses to interactive stories was applied. It includes diverse modes of experience such as suspense, curiosity, flow, and effectance (perceived causal influence onto the story), as well as important preconditions of meaningful experiences such as perceived usability or character believability. For this conceptual framework, the self-report measurement tool provided by VUA (Vermeulen, Roth, Vorderer, & Klimmt, 2010) was used in the present study. It was expanded by the experiential aspect of autonomy (Ryan, Rigby, & Przybylski, 2006), which is particularly relevant to the comparison of continuous versus round-based dialogue: The former might cause stronger perceptions of autonomy, whereas the latter may make users perceive constrained autonomy, as users need to 'wait' until it is their 'turn' to speak during interacting with the story characters (see 1.2 continuous interaction). Moreover, participants were asked about whether the experience had matched their expectations (e.g., "I had expected better graphics") and of which 'conventional' type of (media) experience the interactive story had reminded them. These questions were used to determine whether shifting between dialogue modes affects users' schematic perception of the interactive story (e.g., if the system is experience like a video game or an improvisation theatre).

A total of 42 university students (mean age: 22 years, 30 females) participated in the study. They were randomly assigned to either use the Virtual Beergarden with continuous (n=20) or round-based dialogue (n=22). After receiving a brief introduction of the system, their exposure typically lasted for five to ten minutes. Afterwards, participants filled in a questionnaire about the various dimensions of the user experience.

For all scales of the user experiences, the rating questions on whether expectations were met, and the items about which conventional media experience was perceived to be similar, mean comparison of index or item means between participants exposed to either the continuous or the round based dialogue were computed. T-test statistics were applied to determine group differences of particular importance. Table 1 summarizes selected findings. While in most measured experiential dimensions, user ratings did not differ between the two system versions, some informative group differences emerged. Most importantly, users who interacted with the system through continuous dialogue reported greater perceived autonomy and curiosity, were less disappointed in terms of engagement, and rated the experience more similar to improvisation theatre than the user group confronted with round-based dialogue.

Dimension of user experience	Users confronted with continuous dialogue		Users confronted with round-based dialogue		t-Test
	M	*SD*	*M*	*SD*	*p*
Autonomy	2.68	0.93	2.17	0.79	.055
Curiosity	3.78	0.77	3.33	0.92	.095
Suspense	3.16	0.74	2.82	0.79	n.s.
Enjoyment	3.45	0.95	3.07	0.94	n.s.
Comparisons with expectations					
„I had the expectation that the experience would be more engaging“	1.75	0.97	2.18	0.59	.09
Similarities to other experiences					
“The experience reminded me of playing a video game”	4.25	1.33	4.82	0.39	.06
“The experience reminded me of watching a movie”	2.45	0.99	1.77	1.15	.049
“The experience reminded me a little of playing improvisation theatre”	2.10	1.33	1.36	0.49	.02

Table 1: Selected findings from exploratory user study analysed by IRIS partners from VUA.

The exploration of user responses to two different design options in interactive storytelling – in the present case: options of dialogue management in – has revealed important insights for further system development. First, findings indicate that shifting between technologically quite different options does not affect user experiences in fundamental way: Many conceptually relevant dimensions of the user experience were found to be equal in both experimental groups, and only a few (nearly) significant differences emerged. These differences are highly interesting, however: They suggest that users value continuous dialogue, which comes with greater technological requirements, in terms of greater autonomy and due to greater curiosity about how the story will evolve. Moreover, users' comparisons of the interactive story with previous media experiences shift if the dialogue mode is changed: Continuous dialogue is perceived to be closer to film and improvisation theatre experiences, whereas users judge round-based dialogue to be more similar to video game play, probably to classic menu-based adventure games. So overall, the technologically more ambitious design option of continuous dialogue seems to contribute to a more unique, novel kind of user experience, whereas the less demanding option of round-based dialogue directs users' perceptions towards well-known experiences of interactive entertainment.

The observation of rather small group differences needs some methodological reflections. First, user ratings were obtained from a prototype system that provides a rather short and simple narrative. The fact that within such a brief and not very complex context, dialogue design makes a small yet measurable empirical difference suggests that with a fully developed system (e.g., a Virtual Beergarden soap story with 10 characters and elaborated plot lines), the impact of dialogue options on user experience would be much stronger and profoundly relevant to the overall user experience. In this sense, the 'early' user involvement

turns already out to be valuable. Second, the present study has only compared manifestations of different design options that related to one specific question, namely dialogue management. In synthesizing a complete interactive storytelling system, many more of these design decisions need to be made. So if system creators could make each of the most important of these decisions based on empirical results like the those obtained from the current study, they could harmonize the system components towards a truly optimized end-user experience. The relevance of empirical findings on one particular design option thus should be 'multiplied' across the many issues system inventors have to consider in interactive storytelling to acknowledge the importance of the rather small group differences found in the present experiment.

At the most abstract level, the findings from the current study nicely support our initial argument that early user involvement in the design process of interactive storytelling systems is highly useful. When open design decisions can be broken down into comparisons of distinct options such as the present case of continuous versus round-based dialogue management, systematic user studies that employ a solid theoretical framework and well-construed measures can help to make better-informed design choices. Concerning the question at hand, designers now can discuss whether they want to provide more perceived autonomy and more of an improvisation theatre kind of experience to users (Tannenbaum & Tannenbaum, 2008) or whether they strive for an experience similar to playing a video game. Of course, standardized measures do not tell designers the full story; qualitative approaches with single users are equally important. Yet the fact that theory-based, standardized measures such as (Vermeulen, Roth, Vorderer, & Klimmt, 2010) reveal interpretable and relevant effects of design decisions even with 'small' prototype systems clearly indicates that quantitative-experimental approaches in early user research can make important contributions during ongoing system development as well.

3. Conclusions

In this deliverable, we presented an updated version of the technical framework provided in Deliverable D6.3. The major changes included extensions of the natural language dialogue components, in particular the SceneMaker tool for dialogue management and the semantic parser Spin. Furthermore, we developed taxonomy for full body interaction including gestures and postures.

Most of the components of the technical framework have been made publically available to the research community and attracted partially more than 100 downloads. In the last months, we also focused on extensions of the components that help us make them available also to people without a technical background. Among other things, we gave a course on Scenemaker to students of a media management course at the HSRM. Furthermore, the developed technology has been presented to a professional storywriter (Georg Struck) with whom we are currently jointly authoring a story.

Finally, the technical framework has been used to set up controlled user experiments focusing on appropriate styles of interaction in interactive storytelling systems.

References

Beeck, M. v. (1994). A Comparison of Statecharts Variants. *Proceedings of the Third International Symposium Organized Jointly with the Working Group Provably Correct Systems on Formal Techniques in Real-Time and Fault-Tolerant Systems* (S. 128-148). London, UK: Springer-Verlag.

Braun, N. (2003). Storytelling in Collaborative Augmented Reality Environments. *Proceedings of the 11th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG).*

Bunt, H., Schiel, F., & Steffen, A. (2004). The BITS speech synthesis corpus for German. *Proc. 4th Conference on Language Resources and Evaluation (LREC)*, (S. 2091-2094). Lisbon, Portugal.

Cavazza, M., Charles, F., & Mead, S. J. (2002). Interacting with virtual characters in interactive storytelling. *Proceedings of AAMAS '02.*

Dow, S., Mehta, M., Harmon, E., MacIntyre, B., & Mateas, M. (2007). Presence and engagement in an interactive drama. *Proceedings of the SIGCHI conference on Human factors in computing systems*, (S. 1475-1484).

Engel, R. (2005). Robust and Efficient Semantic Parsing of Free Word Order Languages in Spoken Dialogue Systems. *Interspeech 2005 - Eurospeech*, (S. 3461-3464).

Harel, D. (1987). Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming. 8*, S. 231-274. Elsevier North-Holland, Inc. Amsterdam, The Netherlands.

Klimmt, C., Roth, C., Vermeulen, I., Vorderer, P., & Roth, F. S. (in press). Forecasting the experience of future entertainment technology: "Interactive Storytelling" and media enjoyment. *Games and Culture: A Journal of Interactive Media.*

Li, Y. (2010). Protractor: a fast and accurate gesture recognizer. *Proceedings of the 28th international conference on Human factors in computing systems* (S. 2169-2172). New York, NY, USA: ACM.

Mateas, M., & Stern, A. (2003). Façade: An Experiment in Building a Fully-Realized Interactive Drama. *Game Developers Conference: Game Design Track.*

Mehta, M., Dow, S., Mateas, M., & MacIntyre, B. (2007). Evaluating a conversation-centered interactive drama. *6th international joint conference on Autonomous agents and multiagent systems (AAMAS 2007).*

Packard, E. (1986). *Sugarcane Island.* Bantam.

Roth, C., Klimmt, C., Vermeulen, I., & Vorderer, P. (2011). The Experience of Interactive Storytelling: Comparing "Fahrenheit" with "Façade". *10th International Conference on Entertainment Computing (ICEC 2011).*

Ryan, R. M., Rigby, C. S., & Przybylski, A. (2006). The motivational pull of video games: A self-determination theory approach. *Motivation and Emotion*, (S. 347-363).

Sali, S., Wardrip-Fruin, N., Dow, S., Mateas, M., Kurniawan, S., Reed, A. A., et al. (2010). Playing with words: from intuition to evaluation of game dialogue interfaces. *5th International Conference on the Foundations of Digital Games* (S. 179-186). New York, NY, USA: ACM.

Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., et al. (2011). Real-Time Human Pose Recognition in Parts from Single Depth Images.

Tannenbaum, J., & Tannenbaum, K. (2008). Improvisation and Performance as Models for Interacting with Stories. *First International Conference on Interactive Digital Storytelling (ICIDS 2008)*, (S. 250-263).

Vermeulen, I., Roth, C., Vorderer, P., & Klimmt, C. (2010). Measuring user responses to interactive stories: Towards a standardized assessment tool. *3rd International Conference on Interactive Digital Storytelling (ICIDS 2010)* (S. 38-43). Springer.

Wagner, J., Lingenfelser, F., & André, E. (2011). Social Signal Interpretation (SSI): A Framework for Real-time Sensing of Affective and Social Signals. *KI - Künstliche Intelligenz - Special Issue on Emotion and Computing , 25*.

Walker, M. A., Fromer, J., Di Fabbrizio, G., Mestel, C., & Hindle, D. (1998). What can I say? evaluating a spoken language interface to Email. *SIGCHI conference on Human factors in computing systems* (S. 582-589). New York, NY, USA: ACM Press/Addison-Wesley Publishing Co.

Wobbrock, J. O., Wilson, A. D., & Li, Y. (2007). Gestures without libraries, toolkits or training: a $1 recognizer for user interface prototypes. *20th annual ACM symposium on User interface software and technology* (S. 159-168). New York, NY, USA: ACM.