D5.3: Empathic Tutor Interaction Management System

Work package	Tutor's empathic behaviour and dialogue generation engine (WP5)
Task	Task 5.3
Dissemination Level	Public
Publishing date	Contractual M32 Actual
Deliverable	D5.3
WP / Task responsible	INESC-ID
Contact person	Ana Paiva, INESC-ID
Contributors	INESC-ID, HWU, JacobsUni, UoB, UGOT
Short abstract	The prototype described here is the empathic tutor Interaction Manager (IM) system. We will describe two variants for the two scenarios. The Scenario 1 IM is stochastic in the sense that it will respond with a variety of responses for the same state and interacts with a stochastic user simulator. For Scenario 2, the IM is a combination of rule based and machine learning techniques. Finally, we present a template based natural language generator called Skene generates both verbal and non-verbal behaviours.
Keywords	Interaction Manager, Machine Learning, Natural Language
Documents	Deliverable 5.3

Table of Contents

1. Introduction

This deliverable outlines the prototype described in Task 5.3 of the Description of work. Namely:

- Baseline, non-adaptive Interaction Manager and NLG component prototype
- Combine the interaction management system with a template based natural language generation component (Skene)
- Create student simulations, modelling students with a variety of learning abilities.

As the nature of the interaction for the two scenarios is quite different, we developed two variants of the Interaction Manager described each in turn below. The NLG component (Skene) has been developed to be independent of the scenario and thus provides a common solution for the entire project.

Here, we provide software that demonstrates the functionality of the three prototypes. These demonstrations are available through the following links and using the login details provided below.

https://gaips.tagus.ist.utl.pt:5001/webman/index.cgi
sftp://hwu@gaips.tagus.ist.utl.pt/EmoteShared

Username: emote-reviewer
Password: VwBEpWfB

System Requirements

- Java Runtime Environment JRE 1.7 or above[1].
- Microsoft .NET Framework 4.5[1]
- Microsoft Office Access database engine 2007[1]
- Microsoft Windows 7 or above
- RAM: 512 MB
- Processor: 1 GHz
- Disk Space (with installed libraries):
    - 32 bit processor: 1 GB
    - 64 bit processor: 2.1 GB

---

[1] The installer packages for the required libraries are located into the *"Required Libraries"* folder.

2. Interaction Manager for Scenario 1 prototype presentation

Introduction

We present the demonstration software package for the Interaction Manager (IM) module which is a key decision making module in the robotic tutor system in the context of Scenario 1: Map reading activity. The Interaction Manager module is an interactive module. Hence it is difficult to show how it works in isolation. We have therefore paired it with a User Simulation module that simulates the behaviour of a learner and some other modules of the system in Scenario 1. The user simulation was developed in order to help with the development and testing of the IM module and strategy.

For the prototype demonstration, we present the IM interacting with the user simulation (simulating a learner). Both the IM and the learner behave stochastically. For instance, when presented with a task, the learner could decide to do one of the following actions: answer the task randomly, choose to answer correctly or incorrectly or simply wait. This non-deterministic response triggers different responses from the IM. The IM strategy is also stochastic in the sense that it will respond with a variety of responses for the same state on different runs. Therefore, every instance of the interaction will be different. For a full description of the various modules, scenarios and system architecture, please see Deliverable 6.1.

Instructions

1. Run run-scenario1-im-demo.bat.

2. During execution, the demo will wait for the reviewer to press the return key to proceed. This design was used to give the reviewer a chance to step through the interaction and examine the output in detail (See Figure 1).

Figure 1 Terminal output when running the Scenario 1 IM demonstration

How to interpret the output

The output of the demo is tailored to highlight the workings of the Interaction Manager and User Simulation.

The tutor initiates the conversation by greeting the user. It then introduces the task and moves on presenting the activity one task at a time. This is done through the communicative function *presentTask.* The following output shows the tutor presenting a task to the user.

Tutor CF >> *presentTask*

Tutor DA >> *{"previousSymbol":"InformationCentre","learnerId":"123","distanceSkillRequired":"true",
"toolSkillRequired":"false","correctDistance":50,"toModule":"user","correctTool":"null",
"learnerName":"John","scenarioName":"bristolShort","time-out":-1,"learnerGender":"M",*

"correctDirection":"north","utterance":"taskStep:bristola1s1","communicativeFunction":"presentTask",

*"correctSymbol":"Telephone","distanceMetric":"m",
"symbolSkillRequired":"true","currentStepId":"2","directionSkillRequired":"true"}*

Tutor says >> *<GAZE(student)>Can you tap on the telephone symbol 50 meters north of the information centre?<GAZE(clicks)>*

The first line presents the communicative function (CF) of the Interaction Manager (marked *"System CF >>"*). Communicative Function summarises the intent of the IM. In this case, it wants to present a task step to the user. The communicative function is a part of the IM output which is a dialogue action (DA). The second line in the above output shows the IM's dialogue action (marked *"System DA >>"*). This contains the CF along with all the other parameters necessary to render the intent of the IM into an utterance. An utterance is words and/or gestures to be realised by the Robot. The translation of the dialogue action to utterance is done by Skene, the natural language generation module. Skene first converts the dialogue action into a string of words embedded with gesture mark-ups. We have made this also available in the output above for clarity of context (see third line marked *"System says >>"*). However, the IM sends only the DA to Skene. See the Section 4 for more details on Skene.

As mentioned earlier, we have used a User Simulation module to simulate the behaviour of a learner and other modules that serve as the environment to the Interaction Manager. This module takes as input the IM dialogue action and responds with its own dialogue action. The following output snippet illustrates a learner output.

Learner task response >> *correct*

Learner DA >> *{"actualDistance":50,"distanceCorrect":"true","dirToolUsed":"true","skillLevelSymbol":"high",*

"responseCorrect":"true","symToolUsed":"true","actualDirection":"north","toolCorrect":"true",

"actualTool":"null","disToolUsed":"true","actualSymbol":"Telephone","skillLevelDirection":"low",

"fromModule":"user","symbolCorrect":"true","skillLevelDistance":"medium",

"communicativeFunction":"answerTask","directionCorrect":"true","currentStepId":"2"}

Tutor CF >> *positiveFeedback*

Tutor DA >> *{"previousSymbol":"starting point",*
"learnerId":"123","correctDistance":0,"responseRequired":"false","toModule":"user","learnerName":"John","scenarioName":"bristolShort","time-out":10,"learnerGender":"M","correctDirection":"null",

"communicativeFunction":"positiveFeedback","correctSymbol":"InformationCentre","distanceMetric":"m"}

Tutor says >> *<GAZE(student)> <Face(happiness)> <HEADNOD(1)> Yeah. <GAZE(clicks)>*

A learner in this scenario responds to the tutor by answering the task. He/she does so by touching the correct feature on the map displayed on the touch table. This input is interpreted by the learner model module as described in Deliverable 4.2. Here, we emulate the output of the learner model module in simulation. The first line in the above output presents the key variable of the user's response: whether the user's response to the task is correct or not. This is marked *"Learner task response >>"*. This is extracted from the learner's dialogue action in the next line of the output. This contains all the other parameters that are used by the tutor to evaluate the next pedagogical move. In the above example, the learner gets the answer right and in turn the tutor responds with a positive feedback.

The following example shows another instance, where the learner gets it wrong. The tutor responds with a pedagogical move, *keyword*.

Learner task response >> *incorrect*

Learner DA >> *{"actualDistance":50,"distanceCorrect":"true","dirToolUsed":"true","skillLevelSymbol":"high",*

"responseCorrect":"false","symToolUsed":"true","actualDirection":"north","toolCorrect":"true",

"actualTool":"null","disToolUsed":"true","actualSymbol":"Telephone","skillLevelDirection":"low",

"fromModule":"user","symbolCorrect":"false","skillLevelDistance":"high",

"communicativeFunction":"answerTask","directionCorrect":"false","currentStepId":"2"}

Press return key..

Tutor CF >> *keyword:distanceDirectionSymbol*

Tutor DA >>
{"learnerName":"John","previousSymbol":"InformationCentre","scenarioName":"bristolShort","time-out":-1,"learnerGender":"M","learnerId":"123","correctDistance":50,"correctDirection":"north",

"correctSymbol":"Telephone","communicativeFunction":"keyword:distanceDirectionSymbol",

"toModule":"user","distanceMetric":"m"}

Tutor says >> *<GAZE(student)> <Face(neutral)> 50 meters north Telephone*

Sometimes the learner can take a while to respond to the task. However, the IM does not wait forever and for that reason, it receives a time out message that triggers it to act in the context of elapsed time. The following output shows how this is simulated in the demo.

Learner DA >> {"time-out":"true","fromModule":"hub"}

When a task is finished, the IM communicates with a module called Scenario Manager to receive the next task. This is shown in the following output. The Scenario Manager reads the scenario file and presents the task details to the IM. This is then presented to the learner subsequently. The task utterance identified here as "taskStep:bristola1s2" and is fetched by Skene from the utterance library. Other parameters from the Scenario Manager are used by the IM to identify the pedagogical moves and parametrize the dialogue actions.

Scenario Manager >> {"symbol":"true","direction":"true","tool":"null","step-speech":"taskStep:bristola1s2",

"distance-required":100,"tool-required":"false","direction-required":"east","objectPlacementTask":"false","distance":"true","info-on-completion":"false","time-out":-1,"distance-metric":"m",

"symbol-name-required":"Museum","fromModule":"stm","scenario-name":"bristolShort","stepId":"3"}

Tutor CF >> presentTask

Tutor DA >> {"previousSymbol":"InformationCentre","learnerId":"123","distanceSkillRequired":"true",

"toolSkillRequired":"false","correctDistance":50,"toModule":"user","correctTool":"null",

"learnerName":"John","scenarioName":"bristolShort","time-out":-1,"learnerGender":"M",

"correctDirection":"north","utterance":"taskStep:bristola1s1","communicativeFunction":"presentTask",

"correctSymbol":"Telephone","distanceMetric":"m","symbolSkillRequired":"true","currentStepId":"2",

"directionSkillRequired":"true"}

Tutor says >> <GAZE(student)>Can you tap on the telephone symbol 50 meters north of the information centre?<GAZE(clicks)>

Finally, the input from the affect perception module is also simulated to show how the IM's behaviour changes. The affect perception module outputs the valence and arousal levels of the learner (see Deliverable 6.1 for details). The simulated output of this module is shown below. The example below also shows the tutor reacting to the affect input with small talk ("Is the table difficult to use, you think?").

Affect >> {"arousal":"negative","valence":"negative","fromModule":"affect","confidence":800}

Press return key..

Tutor CF >> smalltalk

Tutor DA >>
{"learnerName":"John","previousSymbol":"InformationCentre","scenarioName":"bristolShort","time-out":"-1", "learnerGender":"M","learnerId":"123","correctDistance":50,"correctDirection":"north", "correctSymbol":"Telephone","communicativeFunction":"smalltalk","toModule":"user","distanceMetric":"m"}

Tutor says >> <GAZE(student)> <Face(neutral)> <GLANCE(throughMap)> Is the table difficult to use, you think?

---

3. Interaction Manager for Scenario 2 prototype presentation

The Scenario 2 Interaction Manager consists of two modules: Logic Web and PBot. Together they manage the interaction with the two users, firing events which lead the system to perform the required utterances and game moves (see Deliverable 6.1 for details of these and other system modules).

Logic Web

The Logic Web is a rule based system, where rules are designed as *cases*. A case is a set of conditions that must be met to activate the case itself. Each condition is a higher level elaboration of the raw messages coming from the Thalamus character.

The left side of the window of Figure 2 shows the list of the cases that are managed by the Logic Web. Each case's name describes the behaviour that will be performed when the conditions for that case are all active.
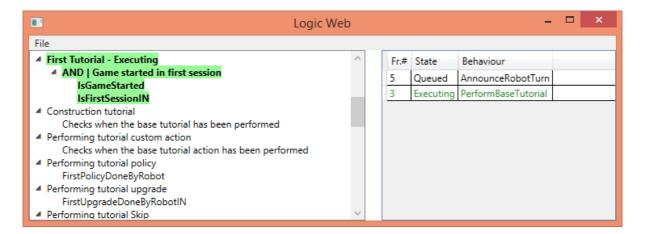

Figure 2 Logic Web Interface

When a condition is active, it becomes highlighted in green. If the case is fired, it is highlighted in green as well and a message relative to its state is added next to it (in the example screenshot it says "executing" next to "First Tutorial"). The right side of the window presents the queue of the behaviours that have been fired. Only one behaviour at a time can be executed while the rest will wait until the current execution ends. Executed behaviours will be shown in light grey, to show a history of the past actions.

PBot

The PBot core is a classifier that listens for messages from the Thalamus character and tries to mimic the behaviour of a human wizard. This human wizard's behaviour was recorded during a

Wizard of Oz experiment run in September 2014. The only task of this module is to fire a request to perform specific utterances when the classifier recognizes a pattern.

The frequency with which the PBot can fire utterances is limited depending on the current state of the system. The PBot can request an utterance only if no utterance was performed in the last 7 seconds. This simple solution avoids having the PBot asking for utterances too often, which would be detrimental to the interaction quality.

Prototype Demonstration

Similarly for Scenario 1, we present here a demonstration of the Interaction Manager for Scenario 2 in simulation. The system comprises of three Thalamus modules, PBot, Logic Web and ThalamusLogPlayer and a Thalamus character (see Deliverable 6.1 for further details of these modules).

The ThalamusLogPlayer is a utility that allows one to "replay" a game session, injecting into the Thalamus character all the messages sent from the modules that were active during the session. By doing so, we can simulate Thalamus events that are happening in modules that, in reality, are not running. In addition, replaying a game session allows us to show how the Logic Web and the PBot behave during a game session.

To facilitate this simulation, the Logic Web have been slightly altered. In the complete system, when the Logic Web fires a behaviour requesting an utterance to be executed, it adds a unique ID to that utterance and waits for that utterance with that ID to finish. In the simulation these IDs don't match so Logic Web ignores the ID of an utterance and accept the *UtteranceFinished* message as the most recently fired utterance that is finished.
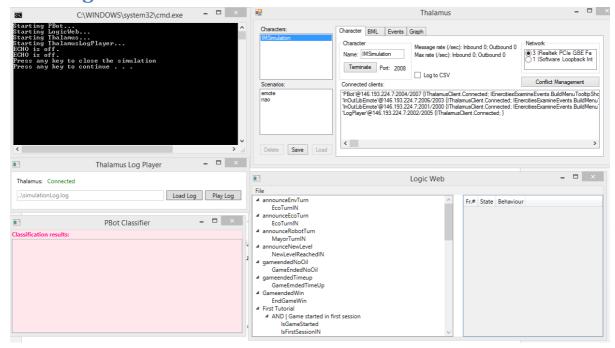
Running the Demo

Figure 3 Demonstration Interface for running the Scenario 2 IM demonstration

To run the demo, first execute the *"Start Prototype Demo.bat"* file in the *"Template Based NLG.zip"* archive. This will run all the modules and the *ThalamusStandalone* required to run the demo (see Figure 3). To start the demo, select the *ThalamusLogPlayer* window and press *Play Log*. This will start the re-play of a pre-recorded session.

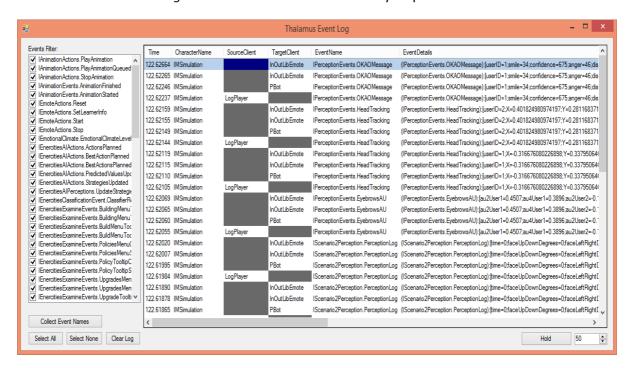Select the *PBot* and the *Logic Web* windows to check how they respond to the simulation.

Figure 4 Thalamus Event Log

The Thalamus message history can be accessed by selecting the *ThalamusStandalone* window, click *on the Events tab and then on the Event Log button (See Figure 4).*

To close the demo just select the shell window related to the batch file used to start the demo and press a key.

4. Template based natural language generator prototype (Skene)

Our template-based natural language generator is called Skene. Skene is the central behaviour manager of the system and is used for both scenarios. It provides several features such as: gazing, pointing, animations and utterance management. Figure 5 shows the Skene management console.
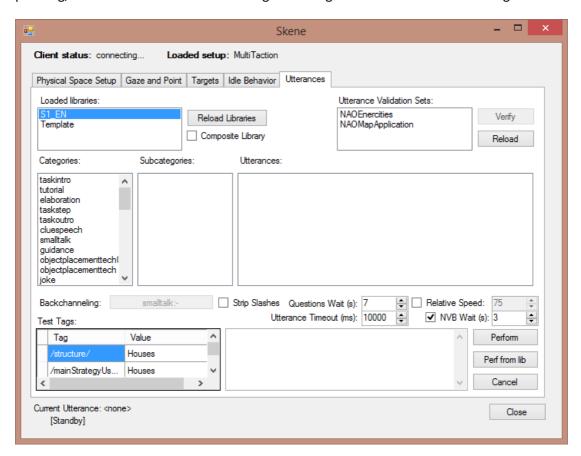

Figure 5 Skene management console

As mentioned above, an utterance is a string containing a phrase in natural language form for the robot's text to speech (TTS) to perform. In addition to the phrase, tags can be added for adjusting the way the TTS realises the phrase, tags to control the robot's behaviour and tags that work as variables that can be substituted by dynamic values when the utterance is performed. An example utterance is given below:

<GAZE(/currentPlayerRole/)> /spd=75/ É a tua vez, /currentPlayerName/.

Where:

- <GAZE(/currentPlayerRole/)> : this tag makes the robot gaze to a target. In this case, the target is a dynamic tag that at runtime will be substituted by the name of the current player's role
- /spd=75/: this is a tag for the robot TTS. In this case, it will cause the TTS to reduce the speed to 75% of the default value.

Prototype Demonstration

As for the IM, we have prepared a simulation environment to demonstrate how Skene works when integrated into the full system. During the demonstration only Skene, the *Logic Web* and the *ThalamusSpeechClient* will be running, along with the Thalamus standalone that creates the Thalamus Character.

The *ThalamusSpeechClient* is a Thalamus module we use for testing. It allows us to perform utterances from Skene using the default Windows TTS engine. Because all the TTS tags present in the utterances are made for the NAO's TTS, they will not be correctly interpreted by Windows TTS, instead they will be read as if they are normal text.

In this demonstration, we are going to show how the flow of messages allow the robot to perform an utterance via the *Logic Web*. Normally, the *Logic Web* continuously listens to Thalamus messages waiting for the right condition to activate a case and consequentially firing a behaviour. In this simulation, the cases will be manually activated. Each case is listed on the left side of the window and showed as a tree. Each sub-item of a case represents a condition required for the case to activate. To manually activate a case, click on all its inner conditions to toggle its active status. Once all the conditions for a case are active, the case itself will become active as well and fire a specific behaviour. All the cases except *"Play action chose by AI"* have a behaviour associated with an utterance, so all of them will send a request to Skene to perform one.

Once the case is active and the behaviour is fired, Skene will receive a request for an utterance of a specific category and subcategory. It will select it and send a *"speak"* message to the Thalamus waiting for a TTS client to perform the phrase.

In this demonstration, the *ThalamusSpeechClient* window will display the text of the performed utterance as well as speaking the phrase.

NOTE: this demonstration described here is for Portuguese therefore the user has to set the Windows TTS language to Portuguese. If the Windows TTS is set to use English language, it will say Portuguese phrases reading them as if they were made of English words. This won't cause any issue to the rest of the system that will keep working as normal.

Running the Demo

To run the demonstration, run the *"Start Prototype Demo.bat"* file in the "Template Based NLG demo.zip" archive. This will run all the modules and the *ThalamusStandalone* needed to run the demonstration. To test the Template-Based Natural Language Generator module (Skene), manually activate the cases in the *Logic Web* so that they fire a request for an utterance that Skene will perform using the *ThalamusSpeechClient* module (see Figure 6).
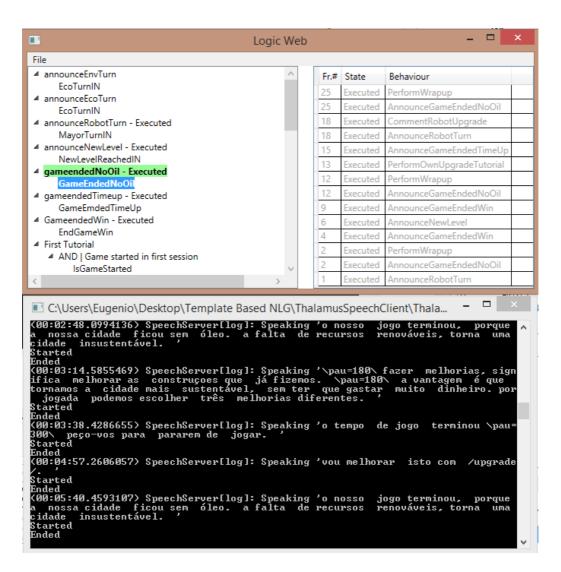
Figure 6 Skene demonstration interface

5. Conclusion

In this deliverable, we have presented demonstration prototypes of interaction management and behaviour management modules and provided instructions for running them. The demos illustrate the functionalities the Interaction Manager and Skene in the working system for Scenario 1 and Scenario 2. We have also shown two user simulations and how these can be used for testing the Interaction Manager.

6. Related Publications

- Ribeiro, T., et al. "From Thalamus to Skene: High-level behaviour planning and managing for mixed-reality characters." Proceedings of the IVA 2014 *Workshop on Architectures and Standards for IVAs*. 2014.
- Srinivasan Janarthanam, Helen Hastie, Amol Deshmukh, Ruth Aylett and Mary Ellen Foster, "A Reusable Interaction Management Module: Use case for Empathic Robotic Tutoring". Proceedings of the 19th workshop on Semantics and Pragmatics of Dialogue, Gothenburg. 2015.