Second Language Tutoring using Social Robots

# Project No. 688014

# L2TOR

## Second Language Tutoring using Social Robots

Grant Agreement Type: Collaborative Project
Grant Agreement Number: 688014

# D3.1 L2TOR system for number domain

Due Date: **30/06/2017**
Submission Date: **07/07/2017**

Start date of project: **01/01/2016** Duration: **36 months**

Organisation name of lead contractor for this deliverable: **ALD**

Responsible Person: **Jean-Marc Montannier** Revision: **1.0**

| **Project co-funded by the European Commission within the H2020 Framework Programme** | | |
|---|---|---|
| **Dissemination Level** | | |
| **PU** | Public | **PU** |
| **PP** | Restricted to other programme participants (including the Commission Service) | |
| **RE** | Restricted to a group specified by the consortium (including the Commission Service) | |

| | | |
|---|---|---|
| **CO** | Confidential, only for members of the consortium (including the Commission Service) | |

# Contents

# Executive Summary

This deliverable is a “demonstrator”. It consists of a prototype comprised of software and hardware, which was built to teach children mathematical concepts in a second language.

## Technical Components

The software developed for the demonstrator is divided in multiple modules interacting as depicted in Figure 1. We will briefly describe the role of each module in this section.

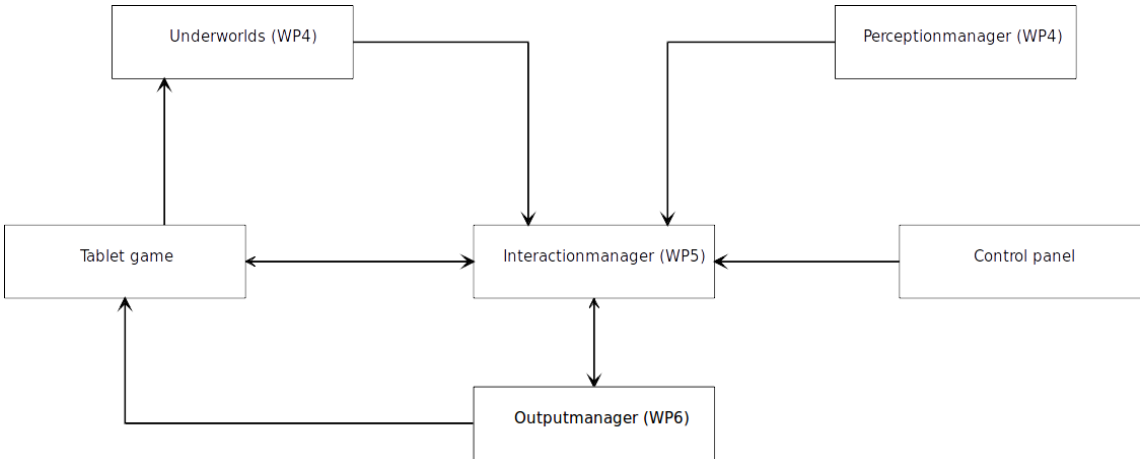


Figure1: Interconnections between the technical modules.

*Tablet Game:* This game supports the interaction between the child and the robot. For the number domain, the game consists mainly in moving objects (e.g. animals) from one place to another (e.g. cages, lakes). This game is implemented as a dynamic HTML page, reacting to information from the *OutputManager* and the *InteractionManager*.

*PerceptionManager:* In the current demonstration, this module is figuring out if the child has spoken. This audio is recorded using either the Kinect device or the tablet microphone and processed using OpenSMILE. Once computed, it is transmitted to the *InteractionManager* for further processing. The *PerceptionManager* is developed in C#.

*Underworlds:* This module receives pre-processed input from the *Tablet Game* and deduces the relation between physical objects. For example, it determines if an elephant is near a cage or in a cage. *Underworlds* is developed in C#.

*Interaction Manager:* This module parses the scenario of a given lesson and makes sure that this scenario is followed by the child. To do so, it takes into account input information (coming from *PerceptionManager, Underworlds* and *ControlPanel*) and decides on the next output (display on the *TabletGame* and *behaviour* to be executed by the *OutputManager*). The *InteractionManager* is developed in C#.

*OutputManager:* This module receives requests from the *InteractionManager*, and ultimately makes the robot speak and move. The *OutputManager* is developed in Python.

*ControlPanel:* The *ControlPanel* offers a graphical interface so that the teacher can have some control over the lessons. In its current implementation, it allows the teacher to start the lesson, and visualize the status of other modules (ON or OFF). The following features will be developed in future version: pause, stop, and step forward. The *ControlPanel* is developed in C#.

*CommunicationManager:* This module is in charge of the communication between the modules. Due to its purely technical nature it does not appear in the Figure 1. All other modules communicate directly to and only with this module. The *CommunicationModule* then parses the received messages in order to forward them to the corresponding modules. The *ConnectionManager* is developed in Python.

## Design decisions

The current architecture in modules communicating through the *Communication-Manager* follows two constraints:

**The system should scale easily to different domains and lessons**. If a lesson requires new functionalities, they can be implemented in the corresponding module without affecting the others. For example, if a new spatial relation is needed most of the work will be done in the *Underworlds* module, without affecting the PerceptionManager or the *OutputManager.*

**It should be easy to work in parallel on multiple aspects.** During the development multiple functionalities are implemented in parallel by various partners. The structure in modules allows to improve certain part of the demonstration without affecting the other ones. The use of a central *CommunicationManager* allows to conduct this development without worrying about a module breaking its communications stack.

The partners are using a git server to share the source code of all modules. Two code camps have been organised in order to promote cooperation between partners and speed-up design choices. A slack chat room has also been created in order to communicate quickly and efficiently after the code camps.

## Results

A video showing the latest stage of development can be found at the following URL: https://uni-bielefeld.sciebo.de/index.php/s/0jXb2lM2ZfujUZL. In the video one can see a test of the number domain with english as the L1 and german as the L2. The demonstrator has to manipulate the animals as instructed by NAO.

Figure 2 shows photos of the pilot study organised at Plymouth University in July 2017. During the pilot study 5 children, aged 5 to 6, interacted with the integrated system.

Additional video of the pilot studies are available, but can for privacy reasons not be shared in this public deliverable.

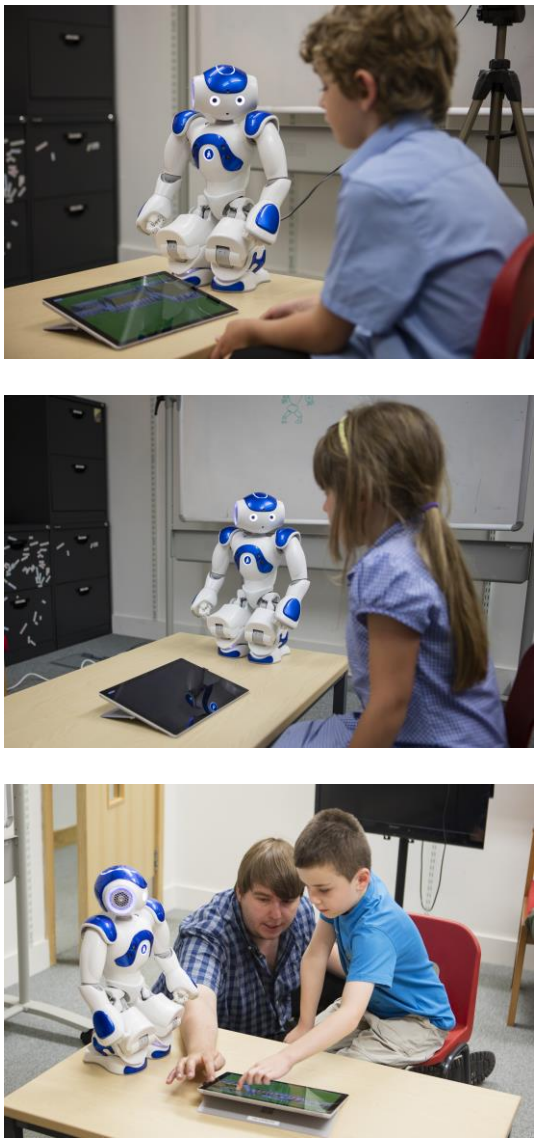

Figure 2: images from the pilot study of the system run at Plymouth University in July 2017.

# Principal Contributors

Jean-Marc Montanier
Fotios Papadopoulos
Christopher Wallbridghe
Mirjam de Haas
Thorsten Schodde
Jan de Wit
Bram Willemsen
Daniel Hernandez
Tony Belpaeme

# Revision History

**Version 1.0 (JMM 12/07/2017)**
First version.

**Version 1.1 (TB 18/07/2017)**
Added information on pilot study.