Tutorial: UML State Machines and Nao

Jérémie TATIBOUET, Shuai LI and François TERRIER

Purpose

This lab is divided into two exercises. These exercises will help you to understand the basics about the usage of UML state machines to specify system behaviors.

Lab N°1 – Nao First Application

Objectives

1. Learn how to use states and transitions.
2. Learn how to use triggers to react on received events.

Setup

1. Import the project "**Lab1-HelloWorldApplication-Student.zip**" into your workspace.
2. Open the composite structure diagram "**HelloWorldSystem**".

Project Description

Structure

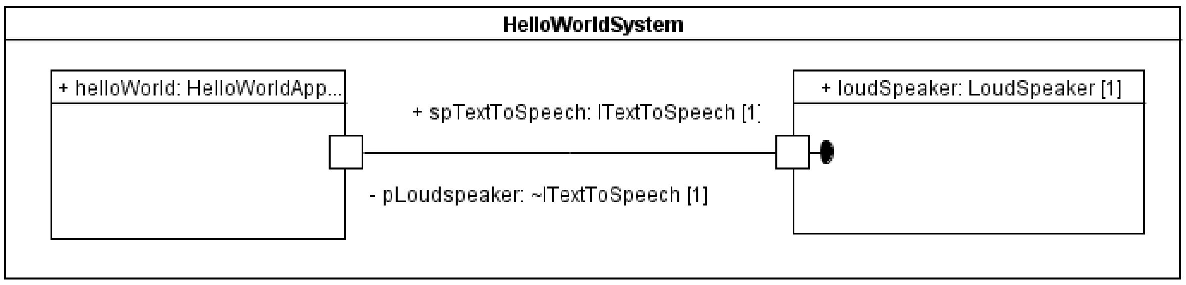
HelloWorldSystem
+ helloWorld: HelloWorldApp...
+ spTextToSpeech: ITextToSpeech [1]
- pLoudspeaker: ~ITextToSpeech [1]
+ loudSpeaker: LoudSpeaker [1]

Figure 1 - Application Structure

The application is composed of two parts typed with actives classes: **HelloWorldApp** and **LoudSpeaker**. Parts can communicates through the connector linking ports **pLoudSpeaker** and **spTextToSpeech**. These ports are typed by interfaces that identify messages (operation calls and signals) that may be passed from one part to another. In this configuration, the operation **say** is *required* by **pLoudSpeaker**. This implies that the part owning the port at the other end of the connector shall *provide* an implementation for the operation **say**. Hence, LoudSpeaker implements that operation and makes sure that when such operation call is received the message that is passed as the parameter to operation is read and emitted through Nao's loud speaker.

Behavior

HelloWorldApp and **LoudSpeaker** are active. Hence they shall have an attached classifier behavior. The **HelloWorldApp** classifier behavior has the role to send a call to the operation say to the classifier behavior of the **LoudSpeaker**. When such a call is received by the **LoudSpeaker**, the message passed as parameter to the operation is displayed in the console.

Expectations

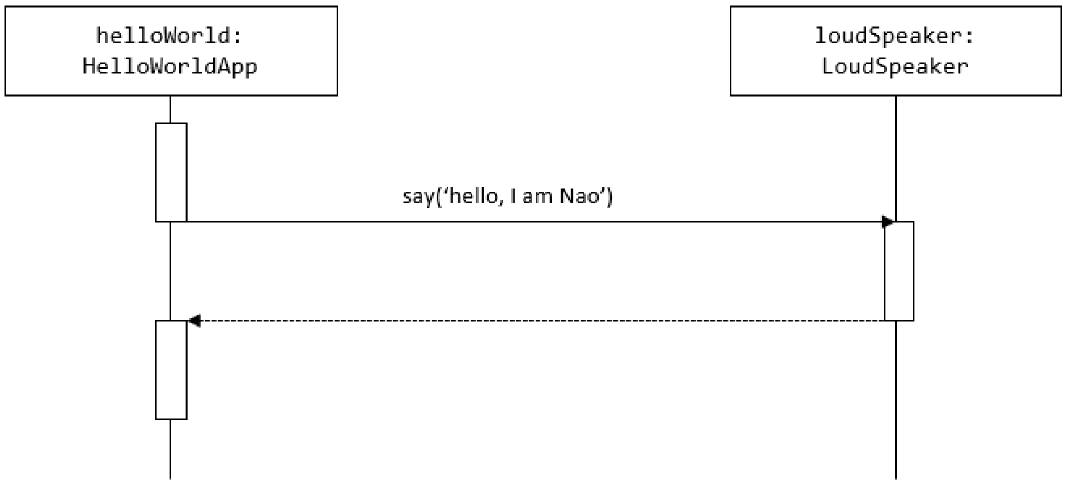

Figure 2 - Nao Audio Application

You must implement the above the specification given by the interaction model depicted in Figure 2.

1. The classifier of **HelloWorldApp** shall be implemented as an activity. This activity shall call in a synchronous way the operation **say**. The operation call shall be emitted through the port owned by the class **HelloWorldApp**.
2. The classifier of **LoudSpeaker** shall be implemented as state machine. This state machine shall provide sufficient information to react upon the reception of a call event for the operation say. As a response to the call the state machine shall display the message passed as parameter to the operation call.
3. The model shall be executable. Any model that is not executable will not be reviewed.

Lab N°2 – Grab Red Ball Application

Objectives

1. Learn to jointly use state machines and activities.
2. Apply what you have learn on state machines and activities to design a complete application enabling Nao to search for red ball and grab it.

Setup

1. Import the project "**Lab2-GrabRedBallApplication-Student.zip**".
2. Open the composite structure diagram "**GrabRedBallSystem**".

Project Description

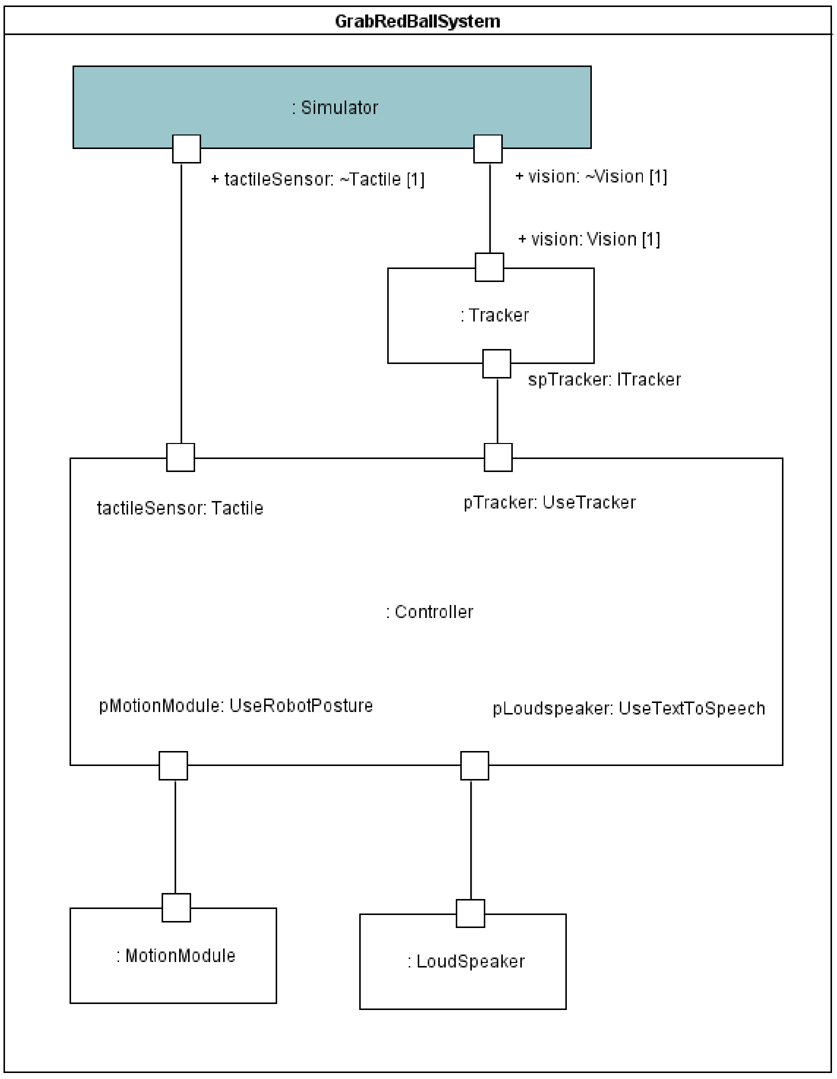

Figure 3 - Grab Red Ball Application

Structure

The system "**GrabRedBallSystem**", is composed of four parts:

1. **Tracker**
   - This component enables Nao to track a specific object in a 3D space. It can receive messages from the simulator and send messages to the controller.
2. **MotionModule**

- This component enables Nao to move in 3D space. It can receive commands from the controller. These commands specify the movements that shall be performed by Nao.

3. **LoudSpeaker**
   - This component enables Nao to express himself through its audio device. Messages to be emitted through the device are specified by the controller.
4. **Controller**
   - This component is the main one of our application. It controls lower level applications such as the tracker, the speaker as well as the tactile sensors.
5. **Simulator**
   - This components emulates the environment of our Nao application. It is typically responsible to emulates human interactions with Nao sensors as well as the presence of objects to track.

Behavior

All classes (**Tracker**, **MotionModule**, **LoudSpeaker**, **Controller** and **Simulator**) involved in this application are active. This imply they all have a classifier behavior.

1. **Tracker**
   - The tracker can receive **ObjectDetected** signals. Such signal have one property that identify the type of object detected in the 3D space. When such object is received a check is performed on the type of an object. If this object is a red ball then a **RedBallDetected** signal is sent to the controller.
2. **MotionModule**
   - The motion module can receive **goToPosture**, **openHand** and **closeHand** operation calls. When such call are received then now actuators are used to make the physical structure of the robot moves
3. **Controller**
   - The controller can receive **TactileSensorPressed** signals from the simulator and **RedBallDetected** signals from the tracker. **TactileSensorPressed** signal enables the receiver to identify the sensor that was pressed. Nao typically has sensors on both hands and head. When Nao head is touched then the tracking of the red ball starts. Tracking continues until the red ball is detected. When detected, the controller makes Nao to open his right hand to grab the ball. The ball is grabbed by Nao when it closes its right hand. This hand can only be closed if the sensor on its right hand is touched.
4. **Simulator**
   - The simulator sends **TactileSensorPressed** signals to the controller. These signals have the role to emulate a stimulation of the head and right hand sensors. The simulator also emulates the presence of objects in front of Nao by sending to the tracker **ObjectDetected** signals.

Expectations

Based on the aforementioned description of the structure and the behavior of the system you must implement:

1. The classifier behavior of the **Tracker** as a state machine.
2. The classifier behavior of the **MotionModule** as a state machine.
3. The classifier behavior of the **Controller** as a state machine.
4. The classifier behavior of the **Simulator** as an activity.
5. The model shall be executable. Any model that is not executable will not be reviewed.

Instructions to hand back your lab

The report (as a PDF) and the models are due by November 20th, 2017.

- Both artefacts will be sent to jeremie.tatibouet@cea.fr and David.Roussel@ensiie.fr
- Please zip your report and your model in an archive "**LAB2-FIRSTNAME-LASTNAME.zip**". Any file that does not match this pattern will not be reviewed.