CSSR for Africa

Culturally Sensitive Social Robotics
for Africa

D3.5 System Integration and Quality Assurance

Due date: **30/06/2024**
Submission Date: **25/07/2024**
Revision Date: **27/01/2025**

Start date of project: **01/07/2023**

Duration: **36 months**

Lead organisation for this deliverable: **Carnegie Mellon University Africa**

Responsible Person: **D. Vernon**

Revision: **1.1**

Project funded by the African Engineering and Technology Network (Afretec) Inclusive Digital Transformation Research Grant Programme		
Dissemination Level		
PU	Public	**PU**
PP	Restricted to other programme participants (including Afretec Administration)	
RE	Restricted to a group specified by the consortium (including Afretec Administration)	
CO	Confidential, only for members of the consortium (including Afretec Administration)	

Executive Summary

Deliverable D3.5 represents the outcome of Task 3.5 and highlights the progress in integrating the software modules critical to the system's development. This deliverable comprises the integration results for four submitted software modules, as well as their corresponding unit test outcomes. The results are based on a checklist designed to assess functionality, performance, and compliance with the system's overall requirements.
The following software modules have been submitted for integration:

D5.2 Animate Behavior Subsystem.

D5.3 Attention Subsystem.

D5.5.1.1 Gesture Execution

D5.5.1.2 Programming by Demonstration.

Each software's integration process and unit test results are documented in dedicated sections within the report, where the corresponding checklist details what was tested for and the results achieved. The following software modules successfully passed the integration tests, meeting all functional requirements and proving their readiness for system deployment:

D5.2 Animate Behavior Subsystem.

D5.3 Attention Subsystem.

D5.5.1.1 Gesture Execution

The following software module is still undergoing integration. Ongoing efforts are focused on addressing outstanding issues to align it with the system's operational expectations.

D5.5.1.2 Programming by Demonstration.

The checklist used in the integration process evaluates the modules against predefined rubrics for functionality and alignment with the software engineering standards set out for the CSSR4Africa project. Each test is marked as passed (✓), failed (✗), or not applicable (−) based on the specific characteristics of the software.
Software modules which pass the integration process have all tests either marked as passed (✓) or not applicable (−), depending on the applicability, while the software modules which fail the integration process have at leats one test marked as failed (✗). The current integration process prioritizes functionality of the software on the physical robot over the simulator robot.

Contents

1 D5.2 Animate Behaviour Subsystem

The **D5.2** Animate Behaviour Subsystem software was submitted for integration on January 10th, 2024. The results indicate the software has passed the integration process. This software module gives the robot the appearance of an animate agent by continually making subtle body movements, flexing its hands a little, and rotating its base slightly.

1.1 Files and Directories

✓ Files for a single component are stored in a subdirectory named **animateBehaviour**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.

✓ The **animateBehaviour** directory has five sub-directories: `config`, `data`, `include/animateBehaviour`, `src`, and `srv`.

✓ The `config` directory contains one file, named.

- ✓ `animateBehaviourConfiguration.ini`
- ✓ The configuration file `animateBehaviourConfiguration.ini` contains the key-value pairs that set the component parameters.
- ✓ Each key-value pair is written on a separate line.

✓ The `data` directory contains three files, named as follows.

- ✓ `animateBehaviourLogFile.log`
- ✓ `pepperTopics.dat`
- ✓ `simulatorTopics.dat`
- ✓ The topic files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
- ✓ Each key-value pair is written on a separate line.

✓ The `include/animateBehaviour` directory contains one file, named:

- ✓ `animateBehaviourInterface.h`

✓ The `src` directory contains two source files, named as follows.

- ✓ `animateBehaviourApplication.cpp`
- ✓ `animateBehaviourImplementation.cpp`

✓ The `srv` directory contains one file, named:

- ✓ `setActivation.srv`

✓ The `animateBehaviour` directory contains a `README.md` file with instructions on how to run the node

✓ The `animateBehaviour` directory contains a `CMakeLists.txt` build file.

✓ The `animateBehaviour` directory contains no `package.xml` manifest file since it is a node within the `cssr_system` package .

1.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

✓ `animateBehaviourApplication.cpp`

```
/* animateBehaviourApplication.cpp
 *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ `animateBehaviourImplementation.cpp`

```
/* animateBehaviourImplementation.cpp
  *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ `animateBehaviourInterface.h`

```
/* animateBehaviourInterface.h
 *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `animateBehaviourApplication.cpp` file contains a documentation comment:

✓
```
/*
 * animateBehaviourApplication.cpp
 * Animate behavior controller for creating lifelike robot movements
 *
 * Implements a ROS node that generates subtle autonomous movements to make the
 * robot appear more lifelike and animate. The system manages three types
 * of movements:
 * - Subtle body joint movements
 * - Hand flexing movements
 * - Small base rotations around the z-axis
 *
 * Key features:
 * - Maintains movements near home positions using randomized patterns
 * - Configurable movement ranges (as percentage of joint ranges)
 * - No head control
 * - Supports selective enabling of different movement types
 * - Can be enabled/disabled via ROS service for social interaction coordination
 * - Configurable via external topic mapping files for physical/simulated robots
 * - Optional verbose mode for movement debugging
 *
 * Debug Settings:
 *   - verboseMode: Enable/disable debug output (default: false)
 *
 *   Range Parameters:
 *   - rotMaximumRange: Maximum rotation range for base movement
 *   - selectedRange: Selected movement range as fraction of maximum
 *   - armMaximumRange: Maximum range for each arm joint [5 values]
 *   - handMaximumRange: Maximum range for hand movement
 *   - legMaximumRange: Maximum range for each leg joint [3 values]
 *
 *   Movement Parameters:
 *   - gestureDuration: Duration of each movement in seconds
```

```
  *   - numPoints: Number of points for arm and hand movements
  *   - numPointsLeg: Number of points for leg movements
  *   - legRepeatFactor: Number of repetitions for leg movements
  *
 *
```

✓
```
 * Libraries:
  *   - ROS core libraries:
  *     - roscpp
  *     - ros/package.h
  *     - actionlib
  *     - control_msgs
  *     - trajectory_msgs
  *     - geometry_msgs
  *   - Standard C++ libraries:
  *     - iostream
  *     - fstream
  *     - thread
  *     - chrono
  *     - vector
  *     - map
  *     - string
  *     - atomic
  *     - random
  *
```

✓
```
  * Parameters:
  *   ROS Parameters:
  *   - None
  *
```

✓
```
 * Configuration File Parameters (animateBehaviourConfiguration.ini):
  *   - platform: Target platform ("robot" or "simulator")
  *   - behaviour: Type of animation behavior ("body", "hands", "rotation", "All")
  *   - simulatorTopics: Topic mapping file for simulator ("simulatorTopics.dat")
  *   - robotTopics: Topic mapping file for robot ("pepperTopics.dat")
  *   - verboseMode: Enable/disable debug output (default: false)
  *   - rotMaximumRange
  *   - selectedRange
  *   - armMaximumRange
      - handMaximumRange
      - legMaximumRange
      - gestureDuration
      - numPoints
      - numPointsLeg
      - legRepeatFactor
  *
```

✓
```
* Subscribed Topics and Message Types:
  *   - None
  *
```

```
✓  * Published Topics and Message Types:
    * - None (This node does not publish any topics directly)
    *
    * Topics Used By Node (But Not Published):
    *   - Action Client Topics (Node sends goals to these action servers):
    *      Physical Robot:
    *           - /pepper_dcm/RightHand_controller/follow_joint_trajectory
    *           - /pepper_dcm/LeftHand_controller/follow_joint_trajectory
    *           - /pepper_dcm/RightArm_controller/follow_joint_trajectory
    *           - /pepper_dcm/LeftArm_controller/follow_joint_trajectory
    *           - /pepper_dcm/Pelvis_controller/follow_joint_trajectory
    *
    *    Simulator:
    *           - /pepper/RightArm_controller/follow_joint_trajectory
    *           - /pepper/LeftArm_controller/follow_joint_trajectory
    *           - /pepper/Pelvis_controller/follow_joint_trajectory
    *
    * Topics Used via Publishers (Node sends messages but doesn't publish topics):
    *    Physical Robot:
    *          - /pepper_dcm/cmd_moveto
    *
    *    Simulator:
    *          - /pepper/cmd_vel
    *
    * Note: It sends action goals and movement commands through established topics
    *  but does not create or publish any topics of its own.
    *

✓  * Services Invoked:
    *  - None
    *

✓  * Services Advertised:
    *   - animateBehaviour/set_activation (cssr_system/set_activation)
    *      Request: string state ("enabled" or "disabled")
    *      Response: bool success
    *

✓  * Input Data Files:
    *   - pepperTopics.dat:
    *   - simulatorTopics.dat:
    *
   *

✓  * Output Data Files:
    *  - animateBehaviourLogFile.log: Log file for runtime messages
    *

✓  * Configuration Files
    *   - animateBehaviourConfiguration.ini:
    *
```

```
✓ *  Example Instantiation of the Module
    *   - rosrun cssr_system animateBehaviour
    *   - rosservice call /animateBehaviour/set_activation "state: 'enabled'"
    *   - rosservice call /animateBehaviour/set_activation "state: 'disabled'"
    *
   *

✓  * Author:  Eyerusalem Mamuye Birhan, Carnegie Mellon University Africa

✓  * Email:   ebirhan@andrew.cmu.edu

✓  * Date:  2025-01-10

✓  * Version: v1.0
```

1.3 Component Unit Testing

The unit testing for the Animate Behaviour Subsystem is packaged in the `unit_tests` package, in the subdirectory `animateBehaviourTest`. The integration report is in section 2 below.

2 Animate Behaviour Subsystem Unit test

The **D5.2** Animate Behaviour Subsystem Unit Test software was submitted for integration on January 10th, 2024. This run the unit test on the `animateBehaviour` node.

2.1 Files and Directories

- ☑ Files are stored in a subdirectory named **animateBehaviourTest**.
- ☑ The **animateBehaviourTest** directory has five sub-directories: `config`, `data`, `include/animateBehaviourTest`, `launch`, and `src`.
- ☑ The `config` directory contains one file, named.
  - ☑ `animateBehaviourTestConfiguration.ini`
  - ☑ The configuration file `animateBehaviourTestConfiguration.ini` contains the key-value pairs that set the component parameters.
  - ☑ Each key-value pair is written on a separate line.
- ☑ The `data` directory contains one file, named as follows.
  - ☑ `animateBehaviourTestOutput.dat`
- ☑ The `include/animateBehaviourTest` directory contains one file, named:
  - ☑ `animateBehaviourTestInterface.h`
- ☑ The `launch` directory contains three files, named as follows.
  - ☑ `animateBehaviourLaunchRobot.launch`
  - ☑ `animateBehaviourLaunchSimulator.launch`
  - ☑ `animateBehaviourLaunchTestHarness.launch`
- ☑ The `src` directory contains two source files, named as follows.
  - ☑ `animateBehaviourTestApplication.cpp`
  - ☑ `animateBehaviourTestImplementation.cpp`
- ☑ The `animateBehaviourTest` directory contains a `README.md` file with instructions on how to run the test.
- ☑ The `animateBehaviourTest` directory contains a `CMakeLists.txt` build file.
- ☑ The `animateBehaviourTest` directory contains no `package.xml` manifest file since it is a node within the `unit_tests` package .

2.2 Internal Source Code Documentation

All source files contain a documentation comment that gives the copyright notice, as follows.

✓ `animateBehaviourTestApplication.cpp`

```
/* animateBehaviourTestApplication.cpp
 *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ `animateBehaviourTestImplementation.cpp`

```
/* gestureExecutionImplementation.cpp
 *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ `animateBehaviourTestInterface.h`

```
/* animateBehaviourTestInterface.h
 *
 * Author: Eyerusalem Mamuye Birhan
 * Date: 2025-01-10
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
```

```
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `animateBehaviourTestApplication.cpp` file contains a documentation comment with the following sections:

✓
```
/*
 * animateBehaviourTestApplication.cpp
 * This code implements a ROS-based test application that uses Google Test
 * to validate the animate behavior module, generates structured test reports,
 * and supports continuous testing with user-controlled iterations.
 *
 * Implements a ROS node that executes a comprehensive test suite for the animate
 * behavior system. The test validates four types of movements:
 * - Subtle body joint movements
 * - Hand flexing movements
 * - Small base rotations around the z-axis
 * - Combined movements of all types
 *
 * Key features:
 * - Continuous test execution capability
 * - Automated test report generation
 * - Configurable test behavior via external configuration
 * - Clean setup and teardown between test runs
 * - User-controlled test repetition
 *
```

✓
```
 * Libraries
 * Libraries:
 *   - ROS core libraries:
 *     - roscpp
 *     - ros/package.h
 *   - Testing libraries:
 *     - gtest/gtest.h
 *   - Standard C++ libraries:
 *     - iostream
 *     - fstream
 *     - ctime
 *     - string
 *
```

✓
```
* Parameters:
 *   - argCount: Number of command line arguments
 *   - argValues: Array of command line argument strings
 *
 * Command-line Parameters:
 *   - Standard ROS parameters
 *   - Google Test command line options
 *
```

✓
```
* Configuration File Parameters:
 *   Test configuration file (animateBehaviourTestConfiguration.ini):
 *   - hands:     True/False    # Enable or disable hand movement tests
 *   - body:      True/False    # Enable or disable body movement tests
 *   - rotation:  True/False    # Enable or disable rotation tests
 *   - All:       True/False    # Enable or disable combined movement tests
 *   Note: Setting value to True runs the test, False skips the test
 *
```

✓
```
* Subscribed Topics and Message Types:
 *   - None directly (handled by test implementations)
 *
```

✓
```
* Published Topics and Message Types:
 *   - None directly (handled by test implementations)
 *
```

✓
```
 * Services Invoked
   *
 *   - /animateBehaviour/set_activation
 *     Used to enable/disable animate behavior during tests
 *
```

✓
```
 * Services Advertised and Message Types
```

✓
```
 * Input Data Files
 *
 * None
 *
```

✓
```
* Output Data Files:
 *   - animateBehaviourTest/data/animateBehaviourTestOutput.dat:Test execution
    results and timing information
```

✓
```
* Configuration Files:
 * - unit_tests/animateBehaviourTest/config/animateBehaviourTestConfiguration.ini:
```

✓
```
 * Example Instantiation of the Module
 *
 * animateBehaviourTestLaunchTestHarness.launch.
 *
 * roslaunch unit_tests animateBehaviourTestLaunchTestHarness.launch
 *
```

✓
```
 * Author:  Eyerusalem Mamuye Birhan, Carnegie Mellon University Africa
```

✓
```
 * Email:   ebirhan@andrew.cmu.edu
```

✓
```
 * Date:  2025-01-10
```

✓
```
 * Version: v1.0
```

2.3 Component Unit Testing

✓ A unit test application named `animateBehaviourLaunchRobot.launch` is provided in the `launch` directory.

✓ A unit test application named `animateBehaviourLaunchSimulator.launch` is provided in the `launch` directory.

✓ A unit test application named `animateBehaviourLaunchTestHarness.launch` is provided in the `launch` directory.

✓ The `animateBehaviourLaunchRobot.launch` file launches the component being tested.

− The `animateBehaviourLaunchSimulator.launch` file launches the component being tested.

✓ The `animateBehaviourLaunchTestHarness.launch` file launches the component being tested.

✓ The `animateBehaviour` being tested outputs the copyright message on startup:

```
\animateBehaviour: v1.0

This project is funded by the African Engineering and Technology Network
 (Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY
```

✓ The `animateBehaviour` being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
\animateBehaviour: start-up.
\animateBehaviour: subscribed to /topicName.
```

✓ The `animateBehaviour` being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
\animateBehaviour: running.
```

✓ The `animateBehaviourLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

− The `animateBehaviourLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

- ✓ The `animateBehaviourLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.
- ✓ Unit test instructions are provided in a file named `README.md` in the `animateBehaviourTest` directory.
  - ✓ The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
  - ✓ The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`.ini`) file are altered.

3 D5.5.1.1 Gesture Execution

The **D5.5.1.1** Gesture Execution software was submitted for integration on January 10th, 2024. The results indicate the software has passed the integration process. This module enables the Pepper robot to execute body and hand gestures. Hand gestures will include deictic, symbolic, and iconic non-verbal gestures. Body gestures will be restricted to bowing and nodding, i.e., lowering and raising gaze.

3.1 Files and Directories

- ✓ Files for a single component are stored in a subdirectory named **gestureExecution**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.
- ✓ The **gestureExecution** directory has six sub-directories: `config`, `data`, `include/gestureExecution`, `msg`, `src`, and `srv`.
- ✓ The `config` directory contains one file, named.
  - ✓ `gestureExecutionConfiguration.ini`
  - ✓ The configuration file `gestureExecutionConfiguration.ini` contains the key-value pairs that set the component parameters.
  - ✓ Each key-value pair is written on a separate line.
- ✓ The `data` directory contains eight files, named as follows.
  - ✓ `gestureDescriptors.dat`
  - ✓ `lArmShakeGestureDescriptors.dat`
  - ✓ `lArmWelcomeGestureDescriptors.dat`
  - ✓ `pepperTopics.dat`
  - ✓ `rArmShakeGestureDescriptors.dat`
  - ✓ `rArmWelcomeGestureDescriptors.dat`
  - ✓ `simulatorTopics.dat`
  - ✓ `waveGestureDescriptors.dat`
  - ✓ The topic files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
  - ✓ Each key-value pair is written on a separate line.
- ✓ The `include/gestureExecution` directory contains two files, named:
  - ✓ `gestureExecutionInterface.h`
  - ✓ `pepperKinematicsUtilitiesInterface.h`

✓ The `msg` directory contains one file, named:

  ✓ `Gesture.msg`

✓ The `src` directory contains three source files, named as follows.

  ✓ `gestureExecutionApplication.cpp`
  ✓ `gestureExecutionImplementation.cpp`
  ✓ `pepperKinematicsUtilitiesImplementation.cpp`

✓ The `srv` directory contains one files, named:

  ✓ `performGesture.srv`

✓ The `gestureExecution` directory contains a `README.md` file with instructions on how to run the node

✓ The `gestureExecution` directory contains a `CMakeLists.txt` build file.

✓ The `gestureExecution` directory contains no `package.xml` manifest file since it is a node within the `cssr_system` package .

3.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

✓ `gestureExecutionApplication.cpp`

```
/* gestureExecutionApplication.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ gestureExecutionImplementation.cpp

```
/* gestureExecutionImplementation.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
* This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ pepperKinematicsUtilitiesImplementation.cpp

```
/* pepperKinematicsUtilitiesImplementation.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
* This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ gestureExecutionInterface.h

```
/* gestureExecutionInterface.h
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ pepperKinematicsUtilitiesInterface.h

```
/* pepperKinematicsUtilitiesInterface.h
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `gestureExecutionApplication.cpp` file contains a documentation comment:

✓
```
/* gestureExecutionApplication.cpp
 *
 * This module is responsible for hosting the service that executes the gestures
    on the robot.
 * The module receives the gesture type, gesture ID, gesture duration,
    bow/nod angle, and the location in the world to pay attention/point to in
    x, y, z coordinates.
 * The module then executes the gesture based on the received parameters.
 *
 * The module supports the execution of deictic, iconic, symbolic, bow, and
    nod gestures.
 * The iconic gestures currently supported are welcome and wave (goodbye) gestures.
```

```
*
* The module also supports the selection of the implementation platform
   (simulator or robot) and the interpolation type (linear or biological motion).
*
* The module is implemented as a ROS service that receives the gesture parameters
* and returns the status of the gesture execution.
*
* The gestures could either be executed using linear velocity interpolation or
   a model of biological motion (minimum-jerk model).
*
* The module subscribes to the /sensor_msgs/joint_states topic to receive
   the joint states of the robot.
* The module also subscribes to the /robotLocalization/pose topic to receive
   the coordinates of the robot in the world.
*
* The module is implemented in C++ and uses the ROS libraries for communication
   with the robot.
* The module is part of the CSSR4A package and is used to execute gestures
    on the robot.
*
*
```

✓

```
* Libraries
* Standard libraries
- std::string, std::vector, std::fstream, std::pow, std::sqrt, std::abs
* ROS libraries
- ros/ros.h, ros/package.h, actionlib/client/simple_action_client.h,
  control_msgs/FollowJointTrajectoryAction.h, geometry_msgs/Twist.h
*
```

✓

```
* Parameters
*
* Command-line Parameters
*
* None
*
```

✓

```
* Configuration File Parameters

* Key                     |     Value
* --------------------- |     -------------------
* platform                    robot
* interpolation               biological
* gestureDescriptors          gestureDescriptors.dat
* simulatorTopics             simulatorTopics.dat
* robotTopics                 pepperTopics.dat
* verboseMode                 true
```

✓

```
* Subscribed Topics and Message Types
*
* /sensor_msgs/joint_states
* /robotLocalization/pose
```

```
✓ * Published Topics and Message Types
   *
   * None

✓  * Services Invoked
    *
   * /overtAttention/set_mode
   *

✓  * Services Advertised and Message Types
    *
   * /gestureExecution/perform_gesture
   *

✓  * Input Data Files
   *
   * pepperTopics.dat
   * simulatorTopics.dat
   * gestureDescriptors.dat
   *

✓ * Output Data Files
   *
   * None
   *

✓  * Configuration Files
   *
   * gestureExecutionConfiguration.ini
   *

✓  * Example Instantiation of the Module
   *
   * rosrun gestureExecution perform_gesture
   *
   * The clients can invoke the service by providing the gesture type, gesture ID,
     gesture duration, bow_nod angle,
   * and the location in the world to pay attention/point to in x, y, z coordinates.
   * The service will execute the gesture based on the received parameters and
     return the status of the gesture execution.
   *
   * Examples of calling the service is shown below:
   * ----- rosservice call /perform_gesture -- deictic 01 3000 25 3.6 2.5 0.82
   * This will execute a pointing gesture with a duration of 3000 ms, and the
     location in the world to point to in x, y, z coordinates.
   *
   * ----- rosservice call /perform_gesture -- bow 01 3000 25 3.6 2.5 0.82
   * This will execute a pointing gesture with a duration of 3000 ms, and bow at an
     angle of 45 degrees.
   *

✓  * Author:  Adedayo Akinade, Carnegie Mellon University Africa

✓  * Email:   aakinade@andrew.cmu.edu
```

- ✓ `* Date:  January 10, 2025`
- ✓ `* Version: v1.0`

3.3 Component Unit Testing

The unit testing for Gesture Execution is packaged in the `unit_tests` package, in the subdirectory `gestureExecutionTest`. The integration report is in section 4 below.

4 Gesture Execution Unit test

The **D5.5.1.1** Gesture Execution Unit Test software was submitted for integration on January 10th, 2024. This run the unit test on the `gestureExecution` node.

4.1 Files and Directories

✓ Files are stored in a subdirectory named **gestureExecutionTest**.

✓ The **gestureExecutionTest** directory has six sub-directories: `config`, `data`, `include/gestureExecutionTest`, `launch`, `src`, and `srv`.

✓ The `config` directory contains one file, named.

- ✓ `gestureExecutionTestConfiguration.ini`
- ✓ The configuration file `gestureExecutionTestConfiguration.ini` contains the key-value pairs that set the component parameters.
- ✓ Each key-value pair is written on a separate line.

✓ The `data` directory contains three files, named as follows.

- ✓ `gestureExecutionTestOutput.dat`
- ✓ `pepperTopics.dat`
- ✓ `simulatorTopics.dat`
- ✓ The topic files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
- ✓ Each key-value pair is written on a separate line.

✓ The `include/gestureExecutionTest` directory contains one file, named:

- ✓ `gestureExecutionTestInterface.h`

✓ The `launch` directory contains three files, named as follows.

- ✓ `gestureExecutionLaunchRobot.launch`
- ✓ `gestureExecutionLaunchSimulator.launch`
- ✓ `gestureExecutionLaunchTestHarness.launch`

✓ The `src` directory contains four source files, named as follows.

- ✓ `gestureExecutionTestApplication.cpp`
- ✓ `gestureExecutionTestDriver.cpp`
- ✓ `gestureExecutionTestImplementation.cpp`
- ✓ `gestureExecutionTestStub.cpp`

- ✓ The `srv` directory contains two files, named:
  - ✓ `setMode.srv`
  - ✓ `setPose.srv`
- ✓ The `gestureExecutionTest` directory contains a `README.md` file with instructions on how to run the test.
- ✓ The `gestureExecutionTest` directory contains a `CMakeLists.txt` build file.
- ✓ The `gestureExecutionTest` directory contains no `package.xml` manifest file since it is a node within the `unit_tests` package .

4.2 Internal Source Code Documentation

All source files contain a documentation comment that gives the copyright notice, as follows.

- ✓ `gestureExecutionTestApplication.cpp`

```
/* gestureExecutionTestApplication.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ gestureExecutionTestImplementation.cpp

```
/* gestureExecutionImplementation.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ gestureExecutionTestInterface.h

```
/* gestureExecutionTestInterface.h
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ `gestureExecutionTestDriver.cpp`

```
/* gestureExecutionTestDriver.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ `gestureExecutionTestStub.cpp`

```
/* gestureExecutionTestStub.cpp
 *
 * Author: Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `gestureExecutionTestApplication.cpp` file contains a documentation comment with the following sections:

✓
```
/* gestureExecutionTestApplication.cpp
 *
 * This module is responsible for running the tests on the gesture execution node.
 * The tests are run using Google Test and the results are written to a file.
 * The module tests the iconic, deictic, bow, nod, and symbolic gestures.
 *
```

```
✓  * Libraries
   * Standard libraries
   - std::string, std::vector, std::fstream, std::pow, std::sqrt, std::abs
   * ROS libraries
   - ros/ros.h, ros/package.h, actionlib/client/simple_action_client.h,
     control_msgs/FollowJointTrajectoryAction.h, geometry_msgs/Twist.h
   *

✓  * Parameters
   *
   * Command-line Parameters
   *
   * None
   *

✓ * Configuration File Parameters

   * Key                     |     Value
   * --------------------- |     --------------------
   * platform                |     robot
   * iconic                        true
   * deictic                       true
   * bow                           true
   * nod                           true
   * symbolic                      false
   *

✓  * Subscribed Topics and Message Types

✓ * Published Topics and Message Types
   *
   * /pepper/cmd_vel                  geometry_msgs/Twist
   *

✓  * Services Invoked
     *
   * /gestureExecution/perform_gesture
   *

✓  * Services Advertised and Message Types

✓  * Input Data Files
   *
   * None
   *

✓  * Output Data Files
   *
   * gestureExecutionTestOutput.dat
   *

✓  * Configuration Files
   *
   * gestureExecutionTestConfiguration.ini
   *
```

✓
```
 * Example Instantiation of the Module
 *
 * rosrun unit_tests gestureExecutionTest
 *
 * The launch file for the gesture execution unit tests is
   gestureExecutionTestLaunchTestHarness.launch.
 *
 * roslaunch unit_tests gestureExecutionTestLaunchTestHarness.launch
 *
```

✓ `* Author:  Adedayo Akinade, Carnegie Mellon University Africa`

✓ `* Email:   aakinade@andrew.cmu.edu`

✓ `* Date:  January 10, 2025`

✓ `* Version: v1.0`

4.3 Component Unit Testing

✓ A unit test application named `gestureExecutionLaunchRobot.launch` is provided in the `launch` directory.

✓ A unit test application named `gestureExecutionLaunchSimulator.launch` is provided in the `launch` directory.

✓ A unit test application named `gestureExecutionLaunchTestHarness.launch` is provided in the `launch` directory.

✓ The `gestureExecutionLaunchRobot.launch` file launches the component being tested.

− The `gestureExecutionLaunchSimulator.launch` file launches the component being tested.

✓ The `gestureExecutionLaunchTestHarness.launch` file launches the component being tested.

✓ The `gestureExecution` being tested outputs the copyright message on startup:

```
\gestureExecution: v1.0

This project is funded by the African Engineering and Technology Network
(Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY
```

- [✓] The `gestureExecution` being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

  ```
  \gestureExecution: start-up.
  \gestureExecution: subscribed to /topicName.
  ```

- [✓] The `gestureExecution` being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

  ```
  \gestureExecution: running.
  ```

- [✓] The `gestureExecutionLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

- [–] The `gestureExecutionLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.

- [✓] The `gestureExecutionLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.

- [✓] Unit test instructions are provided in a file named `README.md` in the `gestureExecutionTest` directory.

  - [✓] The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
  - [✓] The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`.ini`) file are altered.

5 D5.3 Attention Subsystem

The **D5.3** Attention Subsystem software was submitted for integration on January 17th, 2024. The results indicate the software has passed the integration process. This module allows the robot to pay attention to salient features in the environment or to a given location in the environment. The primary salient features are those that are associated with social interaction, mainly people's faces. This feature is also the focus of attention when the robot is not interacting, but is merely observing its environment. As such, attention also contributes to the animate behavior of the robot. Observing the environment also involves scanning the environment for interesting objects which requires the robot to change its focus of attention after some time and not return directly to the original focus of attention.

5.1 Files and Directories

- ✓ Files for a single component are stored in a subdirectory named **overtAttention**. Refer to Deliverable D3.1 System Architecture for details of the ROS package names and the associated ROS nodes.
- ✓ The **overtAttention** directory has six sub-directories: `config`, `data`, `include/overtAttention`, `msg`, `src`, and `srv`.
- ✓ The `config` directory contains one file, named.
  - ✓ `overtAttentionConfiguration.ini`
  - ✓ The configuration file `overtAttentionConfiguration.ini` contains the key-value pairs that set the component parameters.
  - ✓ Each key-value pair is written on a separate line.
- ✓ The `data` directory contains two files, named as follows.
  - ✓ `pepperTopics.dat`
  - ✓ `simulatorTopics.dat`
  - ✓ The topic files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
  - ✓ Each key-value pair is written on a separate line.
- ✓ The `include/overtAttention` directory contains one file, named:
  - ✓ `overtAttentionInterface.h`
- ✓ The `msg` directory contains two files, named:
  - ✓ `Mode.msg`
  - ✓ `Status.msg`

- ✓ The `src` directory contains two source files, named as follows.
  - ✓ `overtAttentionApplication.cpp`
  - ✓ `overtAttentionImplementation.cpp`
- ✓ The `srv` directory contains one file, named:
  - ✓ `setMode.srv`
- ✓ The `overtAttention` directory contains a `README.md` file with instructions on how to run the node
- ✓ The `overtAttention` directory contains a `CMakeLists.txt` build file.
- ✓ The `overtAttention` directory contains no `package.xml` manifest file since it is a node within the `cssr_system` package .

5.2 Internal Source Code Documentation

Refer to Deliverable D3.2, Appendix B (Mandatory Standards for Internal Source Code Documentation), for a definition of the standards on which this checklist is based.

All source files contain a documentation comment that gives the copyright notice, as follows.

✓ `overtAttentionApplication.cpp`

```
/* overtAttentionApplication.cpp
 *
 * Author: Mohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ overtAttentionImplementation.cpp

```
/* overtAttentionImplementation.cpp
 *
 * Author: Mohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
* This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ overtAttentionInterface.h

```
/* overtAttentionInterface.h
 *
 * Author: Mohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `overtAttentionApplication.cpp` file contains a documentation comment:

✓
```
/* overtAttentionApplication.cpp
 *
 * This module module equips the robot with the ability to direct its gaze toward
 *  salient features in its environment  or focus on specific locations,
 * facilitating socially and contextually appropriate behaviors.
 * This capability is crucial for enhancing the robot's ability to interact
 * effectively with people and adapt to dynamic environments.
 *
 * The module operates in five distinct modes, each tailored to a specific context.
 *    - Social mode is activated during social interactions, allowing the robot to
          focus on human faces, and voices.
 *      In this mode, the robot prioritizes social cues to maintain engagement
         and responsiveness.
 *
 *    - Scanning mode, on the other hand, is used when the robot is not engaged in
         social interaction.
 *      In this state, the robot scans its surroundings for potential interaction
         opportunities,
 *      focusing on people and objects of interest while giving higher priority to
         faces.
 *      The robot periodically shifts its focus to new areas,
         ensuring comprehensive environmental coverage.
 *
 *    - In location mode, the robot gazes at a specific target in its environment.
 *      If the robot's head cannot achieve the required pose to fixate
         on the target,
 *      the robot's base rotates to realign its head and body.
 *
 *    - Seeking mode enables the robot to establish mutual gaze with a nearby
         person by searching for a face looking directly at it.
 *      If this process is unsuccessful within a given timeframe, the robot returns
         either a success or failure status.
 *
 *    - Lastly, in disabled mode, the robot's head remains centered and stationary,
         effectively deactivating the attention mechanism.
 *
 * To determine the focus of attention, the module generates two types of saliency
   maps.
 *    - A social saliency map leverages data from face detection and sound
         localization to identify socially significant features.
 *    - A general saliency map, on the other hand, uses information-theoretic
         models to identify visually conspicuous elements
 *      in the robot's environment. These maps form the basis for the robot's
         attentional behavior across different modes.
 *
 * In scanning mode, three key processes work together to enhance
   attentional dynamics.
 *    - First, a winner-take-all (WTA) mechanism identifies a single focus of
         attention from the saliency map using a selective tuning model.
 *    - Second, an Inhibition-of-Return (IOR) mechanism ensures that previously
         attended locations are deprioritized, encouraging exploration of new areas.
```

```
*    - Third, a habituation process gradually reduces the salience of the current
        focus, ensuring that attention does not remain fixated on a single point
       for an extended period.
*
* The robot's gaze is directed by publishing control commands to the
    headYaw and headPitch joints,
* which align the head toward the selected focus of attention.
* For aural attention, the robot adjusts its headYaw angle based on the angle of
   arrival of the sound.
* Calibration parameters ensure accurate mapping between visual offsets in the
   image and the corresponding head joint angles.
* When the required headYaw rotation exceeds a predefined threshold, the module
   coordinates the movement of the robot's base
* and head to maintain focus while realigning the head and torso.
*
* The module's functionality is supported by four key inputs.
* Data from the face detection and sound detection nodes inform the saliency map
   in social mode.
* An RGB image from the robot's camera is used to compute the saliency map in
   scanning mode,
* while the robot's current pose is utilized for attending to specific locations.
* These inputs enable the module to adapt its behavior dynamically based on
   environmental conditions.
*
* The module provides four outputs to facilitate its operation.
* First, it publishes control commands to the robot's headYaw and headPitch
   joints,
* as well as to the wheels and angular velocity when adjusting the robot's pose.
* Second, it generates an RGB image visualizing the saliency function and the
    current focus of attention,
* which can be displayed in verbose mode for debugging purposes.
* Third, the module continuously publishes the current active mode to the
   /overtAttention/mode topic,
* enabling other system components to monitor the robot's attentional state.
* Finally, the module updates actuator topic names based on configuration files
   specific to either the physical robot or a simulation environment.
*
* The module's operation is managed through dedicated ROS services. The module
   advertises services to allow the selection
* of operational modes, such as social, scanning, or location mode.
*
* For ease of analysis, the module can also operate in verbose mode, where
   published data is printed to the terminal,
* and output images are displayed in OpenCV windows.
*
*
```

☑

```
* Libraries
* Standard libraries
- std::string, std::vector, std::fstream, std::pow, std::sqrt, std::abs
* ROS libraries
- ros/ros.h, ros/package.h, actionlib/client/simple_action_client.h,
  control_msgs/FollowJointTrajectoryAction.h, geometry_msgs/Twist.h
```

```
     *
✓    * Parameters
     *
     * Command-line Parameters
     *
     * The attention mode to set the attention system to
     * The location in the world to pay attention to in x, y, z coordinates
     *
✓   * Configuration File Parameters

     * Key                    |     Value
     * --------------------- |     -------------------
     * platform                     simulator
     * camera                       FrontCamera
     * realignmentThreshold         5
     * xOffsetToHeadYaw             25
     * yOffsetToHeadPitch           20
     * simulatorTopics              simulatorTopics.dat
     * robotTopics                  pepperTopics.dat
     * verboseMode                  true

✓    * Subscribed Topics and Message Types
     *
     * /faceDetection/direction              faceDetection.msg
     * /robotLocalization/pose               sensor_msgs::JointState
     * /soundDetection/data                  std_msgs::Float64
     * /naoqi_driver/camera/front/image_raw  sensor_msgs::ImageConstPtr

✓   * Published Topics and Message Types
     *
     * /pepper_dcm/Head_controller/follow_joint_trajectory
     * /cmd_vel
     * /overtAttention/mode

✓    * Services Invoked
     *
     * None
     *

✓    * Services Advertised and Message Types
      *
     * /overtAttention/set_mode
     *

✓    * Input Data Files
     *
     * pepperTopics.dat
     * simulatorTopics.dat
     *

✓   * Output Data Files
     *
     * None
     *
```

✓
```
* Configuration Files
*
* overtAttentionConfiguration.ini
*
```

✓
```
* Example Instantiation of the Module
*
* rosrun cssr_system overtAttention
*
* The clients can call the service by providing the attention mode and the
   location to pay attention to in the world.
* The service will execute the attention mode selected and attend to the location
   provided if mode being set is location mode.
* An example of calling the service is shown below:
* ----- rosservice call /overAttention/set_mode -- location 3.0 2.0 1.0
* This will set the attention mode to location and the location to pay attention
    to is (3.0, 2.0, 1.0)
*
```

✓
```
* Author:  Muhammed Danso and Adedayo Akinade, Carnegie Mellon University Africa
```

✓
```
* Email:  mdanso@andrew.cmu.edu, aakinade@andrew.cmu.edu
```

✓
```
* Date:  January 10, 2025
```

✓
```
* Version: v1.0
```

5.3 Component Unit Testing

The unit testing for Attention Subsystem is packaged in the `unit_tests` package, in the subdirectory `overtAttentionTest`. The integration report is in section 6 below.

6 Attention Subsystem Unit test

The **D5.3** Attention Subsystem Unit Test software was submitted for integration on January 17th, 2024. This run the unit test on the `overtAttention` node.

6.1 Files and Directories

- ✓ Files are stored in a subdirectory named **overtAttentionTest**.
- ✓ The **overtAttentionTest** directory has six sub-directories: `config`, `data`, `include/overtAttentionTest`, `launch`, `msg`, and `src`.
- ✓ The `config` directory contains one file, named.
  - ✓ `overtAttentionTestConfiguration.ini`
  - ✓ The configuration file `overtAttentionTestConfiguration.ini` contains the key-value pairs that set the component parameters.
  - ✓ Each key-value pair is written on a separate line.
- ✓ The `data` directory contains three files, named as follows.
  - ✓ `overtAttentionTestOutput.dat`
  - ✓ `pepperTopics.dat`
  - ✓ `simulatorTopics.dat`
  - ✓ The topic files `pepperTopics.dat` and `simulatorTopics.dat` contain the key-value pairs that set the topic names required by the component.
  - ✓ Each key-value pair is written on a separate line.
- ✓ The `include/overtAttentionTest` directory contains one file, named:
  - ✓ `overtAttentionTestInterface.h`
- ✓ The `launch` directory contains three files, named as follows.
  - ✓ `overtAttentionLaunchRobot.launch`
  - ✓ `overtAttentionLaunchSimulator.launch`
  - ✓ `overtAttentionLaunchTestHarness.launch`
- ✓ The `msg` directory contains one file, named:
  - ✓ `faceDetection.msg`

✓ The `src` directory contains three source files, named as follows.

  - ✓ `overtAttentionTestApplication.cpp`
  - ✓ `overtAttentionTestDriver.cpp`
  - ✓ `overtAttentionTestImplementation.cpp`

✓ The `overtAttentionTest` directory contains a `README.md` file with instructions on how to run the test.

✓ The `overtAttentionTest` directory contains a `CMakeLists.txt` build file.

✓ The `overtAttentionTest` directory contains no `package.xml` manifest file since it is a node within the `unit_tests` package .

6.2 Internal Source Code Documentation

All source files contain a documentation comment that gives the copyright notice, as follows.

✓ `overtAttentionTestApplication.cpp`

```
/* overtAttentionTestApplication.cpp
 *
 * Author: Mohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
     (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ overtAttentionTestImplementation.cpp

```
/* overtAttentionImplementation.cpp
 *
 * Author: Mohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

☑ overtAttentionTestInterface.h

```
/* overtAttentionTestInterface.h
 *
 * Author: AMohammed Danso, Adedayo Akinade
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
   (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

✓ `overtAttentionTestDriver.cpp`

```
/* overtAttentionTestDriver.cpp
 *
 * Author: Mohammed Danso, Adedayo Akinadee
 * Date: January 10, 2025
 * Version: v1.0
 *
 * Copyright (C) 2023 CSSR4Africa Consortium
 *
 * This project is funded by the African Engineering and Technology Network
    (Afretec)
 * Inclusive Digital Transformation Research Grant Programme.
 *
 * Website: www.cssr4africa.org
 *
 * This program comes with ABSOLUTELY NO WARRANTY.
 */
```

The `overtAttentionTestApplication.cpp` file contains a documentation comment with the following sections:

✓
```
/* overtAttentionTestApplication.cpp
 *
 * This module is responsible for running the tests on the overt attention module.
 * The tests are run using Google Test and the results are written to a file.
 * The module tests the scanning, social, seeking, location and disabled modes of
    the overtAttention node
 *
```

✓
```
 * Libraries
 * Standard libraries
 - std::string, std::vector, std::fstream, std::pow, std::sqrt, std::abs
 * ROS libraries
 - ros/ros.h, ros/package.h, actionlib/client/simple_action_client.h,
   control_msgs/FollowJointTrajectoryAction.h, geometry_msgs/Twist.h
 *
```

✓
```
 * Parameters
 *
 * Command-line Parameters
 *
 * None
 *
```

```
✓ * Configuration File Parameters

   * Key                     |      Value
   * ---------------------- |      --------------------
   * platform                         robot
   * scanning                         true
   * social                           true
   * seeking                          true
   * location                         true
   * disabled                         true
   * verboseMode                      true
   *

✓  * Subscribed Topics and Message Types
    *
    * /faceDetection/direction               faceDetection.msg
   * /robotLocalization/pose                sensor_msgs::JointState
   * /soundDetection/data                   std_msgs::Float64
   * /naoqi_driver/camera/front/image_raw  sensor_msgs::ImageConstPtr
   * /overtAttention/mode                   Status.msg

✓ * Published Topics and Message Types
   *
   * None
   *

✓  * Services Invoked
     *
   * /overtAttention/set_mode
   *

✓  * Services Advertised and Message Types
    *
    * None

✓  * Input Data Files
   *
   * pepperTopics.dat
   * simulatorTopics.dat
   *

✓  * Output Data Files
   *
   * overtAttentionTestOutput.dat
   *

✓  * Configuration Files
   *
   * overtAttentionTestConfiguration.ini
   *

✓  * Example Instantiation of the Module
   *
   * roslaunch unit_tests overtAttentionTestLaunchTestHarness.launch
   *
```

✓ ⋆ Author: Muhammed Danso and Adedayo Akinade, Carnegie Mellon University

✓ ⋆ Email: mdanso@andrew.cmu.edu, aakinade@andrew.cmu.edu

✓ ⋆ Date: January 10, 2025

✓ ⋆ Version: v1.0

6.3 Component Unit Testing

✓ A unit test application named `overtAttentionLaunchRobot.launch` is provided in the `launch` directory.

✓ A unit test application named `overtAttentionLaunchSimulator.launch` is provided in the `launch` directory.

✓ A unit test application named `overtAttentionLaunchTestHarness.launch` is provided in the `launch` directory.

✓ The `overtAttentionLaunchRobot.launch` file launches the component being tested.

− The `overtAttentionLaunchSimulator.launch` file launches the component being tested.

✓ The `overtAttentionLaunchTestHarness.launch` file launches the component being tested.

✓ The `overtAttention` being tested outputs the copyright message on startup:

```
\overtAttention: v1.0

This project is funded by the African Engineering and Technology Network
(Afretec)
Inclusive Digital Transformation Research Grant Programme.

Website: www.cssr4africa.org

 This program comes with ABSOLUTELY NO WARRANTY
```

✓ The `overtAttention` being tested writes short messages to the terminal during the start-up phase to indicate the state of the node:

```
\overtAttention: start-up.
\overtAttention: subscribed to /topicName.
```

✓ The `overtAttention` being tested periodically (every ten seconds) writes a short heartbeat message to the terminal to indicate the state of the node:

```
\overtAttention: running.
```

✓ The `overtAttentionLaunchRobot.launch` file connects the component a data source and a data sink on the physical robot, and produces the expected result as set out in the `README.md` file. This means this file launches the physical robot and all its sensors and actuators as required.

- [−] The `overtAttentionLaunchSimulator.launch` file connects the component a data source and a data sink on the simulator. This means this file launches the simulator robot and all its sensors and actuators as required.
- [✓] The `overtAttentionLaunchTestHarness.launch` file connects the component a data source and a data sink using a driver and a stub. This means this file launches the required drivers and stubs, and the node being tested.
- [✓] Unit test instructions are provided in a file named `README.md` in the `overtAttentionTest` directory.
  - [✓] The instructions explain how the communication and computation functionality is validated by describing the (sink) output data that will be produced from the (source) input data.
  - [✓] The instructions explain how the configuration functionality is validated by describing what changes in behaviour will occur if the values for the component parameters in the component configuration (`.ini`) file are altered.

Principal Contributors

The main authors of this deliverable are as follows (in alphabetical order).

Adedayo Akinade, Carnegie Mellon University Africa.
David Vernon, Carnegie Mellon University Africa.

Document History

Version 1.0
First draft.
David Vernon.
25 July 2024.

Version 1.1
Updated the integration results for gestureExecution (Success).
Updated the integration results for animateBehaviour (Success).
Updated the integration results for overtAttention (Success).
Adedayo Akinade.
27 January 2025.