CSSR for Africa

Culturally Sensitive Social Robotics for Africa

Work Plan

Version 3.2
March 25, 2025

Start date of project:	**01/07/2023**
Duration:	**36 months**
Partner organisations:	**Carnegie Mellon University Africa** **The University of the Witwatersrand**

Project funded by the African Engineering and Technology Network (Afretec)
Inclusive Digital Transformation Research Grant Programme

Executive Summary

This document sets out the work plan that will be used to conduct a three-year research project to develop culturally sensitive social robotics for Africa (CSSR4Africa). The objectives of the project are (i) to identify the verbal and non-verbal social and cultural norms of human interaction that are prevalent in countries in Africa, (ii) to encapsulate them in the behavioral patterns of social robots so that they can engage with African people in a manner that is consistent with their expectations of acceptable social interaction, and (iii) to demonstrate these culturally-sensitive social robot behaviors in two use cases: one for giving a tour of a university laboratory, and one for assisting and giving directions to visitors at the reception of a university. In order to ensure that the project objectives can be achieved in the time available, we restrict the scope of the project to the cultures and social practices that are prevalent in Rwanda and South Africa.

Contents

Chapter 1

Project Description

1.1 Motivation

This research project is motivated by the recognition that, for it to be successful, socio-economic development in Africa must be sensitive to people's culture (Olasunkanmi, 2011). Dignum (2023) drives this home when, in the recently published book *Responsible AI in Africa* (Eke et al., 2023), she says "research and development of AI systems must be informed by diversity, in all the meanings of diversity, and obviously including gender, cultural background and ethnicity". While the overarching agenda of the inclusive digital transformation of Africa is widely recognized to have the potential to be a positive disruptive influence on many aspects of the lives of African citizens, the transition from recognition of potential to realization of benefits is not a straightforward matter. The transition depends on transforming technological invention into innovation, a process that depends on widespread adoption. However, adoption, especially of AI, depends on trust (Alupo et al., 2022), which, in turn, depends on sensitivity to social and cultural factors (Lee & See, 2004).

Successful innovation also depends on infrastructure. Rose (2010) notes that "infrastructure is the unnoticed precondition for technology innovation". There are two forms of infrastructure, the physical and the social. The physical infrastructure might include the availability of electrical power, communications networks, or internet connectivity. Of equal importance is the social infrastructure which includes the social conventions that govern people's behavior and the practices they find acceptable and unacceptable. Social infrastructure heavily impacts on whether or not an invention is adopted and becomes an innovation that can yield benefits for the local community. Again, social infrastructure includes trust and people's sense of what is trustworthy.

1.2 Social Robotics

There is an increasing need for artificial intelligence technology that is capable of interacting effectively with humans. This includes social robots which serve people in a variety of ways. The global social robotics market was valued at USD 1.98 billion in 2020 and is expected to reach USD 11.24 billion by 2026, registering a compound annual growth rate (CAGR) of 34.34% during the period of 2021-2026 (Research and Markets, 2022).

Social robots are designed to operate in everyday environments, often in open spaces such as hospitals, exhibition centers, and airports, providing assistance to people, typically in the form of advice, guidance, or information. The people interacting with the robot have no special training and they expect the robot to be able to interact with them on *their* terms, not the robot's. There are two aspects to this expectation.

First, it means that social robots need to be able to interpret the intentions of the people with whom they are interacting. This is difficult to achieve because humans do not necessarily

articulate their specific needs explicitly when they interact with social robots (or, indeed, with other humans). As Sciutti et al. (2018) note, "the ability of the robot to anticipate human behavior requires a very deep knowledge of the motor and cognitive bases of human-human interaction".

Second, and conversely, humans have expectations of the robot's behavior and, specifically, they expect the robot to act in a trustworthy, culturally-sensitive, socially-acceptable manner, and they have a distinct preference for robots that exhibit legible and predictable behavior (Sciutti et al., 2018). Since people make predictions based on what they are used to, this is somewhat easier to achieve, provided the robot behaviors are tuned to the socio-cultural context in which they are operating.

People use spatial, non-verbal, and verbal communication when interacting with other people. So too must social robots, if they are to be effective. However, successful interaction requires acceptance and trust, which depend on social and cultural norms. These norms impact on the nature of the robot's non-verbal and verbal expression as well as its appearance and spatial behavior. Consequently, they determine the acceptance of social robots and the effectiveness of their interaction (Bartneck et al., 2020). While the case for culturally competent robots has been well made (Bruno et al., 2017b; Khaliq et al., 2018), and while there are studies on cultural differences in the acceptance of robots in the West and East, e.g., by Kaplan (2004); Bartneck et al. (2005); Bruno et al. (2017a), similar studies of the cultural factors that impact on acceptance in Africa have not been reported (Bartneck et al., 2020).[1] This situation will be redressed in the research set out in the work plan in Section 2.

1.3 Research Objectives

The successful deployment of social robots in Africa depends on the robots being accepted by African citizens. Culturally sensitive robot behavior, the focus of this project, is a prerequisite for this. As such, this research project is concerned with the second aspect of effective interaction by social robots identified in the previous section: the need for predictable and culturally-acceptable patterns of robot behavior.[2] In other words, the robot must adapt to, or be adapted to the cultural environment. However, rather than attempt to learn these patterns through interaction (a research goal that would involve significantly more effort than is feasible in a project of this size), we aim to identify these patterns through ethnographic research and then embed them in reconfigurable and reusable interaction primitives that can be utilized when developing the interaction behaviors for the application and environment at hand. Thus, we aim to identify the interaction patterns that are socially and culturally acceptable in Africa, and the specific behaviors that will make social robots capable of courteous, respectful, and engaging interaction by reflecting the social and cultural norms of African people.

The factors that underpin effective human-robot interaction include spatial interaction (proxemics, localization and navigation, socially appropriate positioning, initiation of interaction, communication of intent), nonverbal interaction (e.g., gaze and eye movement, deictic, iconic, symbolic, and beat gesture, mimicry and imitation, touch, posture and movement, and interaction rhythm and timing), and verbal interaction (e.g., speech, speech recognition, language understanding, speech generation) (Bartneck et al., 2020). These spatial, nonverbal, and

[1] The recent survey by Lim et al. (2021) briefly mentions Egypt, Tunisia, Libya, and Sudan but only to contrast perceptions with the Gulf region when interacting with an Arabic robot.

[2] The ability to interpret the verbal and non-verbal expressions of humans is a major challenge in human-robot interaction. Since the development of capabilities for such natural interaction would require much more time and effort than can be accommodated in a project the size of CSSR4Africa, we restrict most interaction by the human to simple requests using the tablet PC on the Pepper robot. In anticipation of more natural interaction in the future, we include a task to address automated speech recognition for Kinyarwanda speaker but we do not address the interpretation of the human's gestures and communication behaviors.

verbal interaction factors must reflect the cultural knowledge that would make social robots acceptable in Africa.

This cultural knowledge will also be used to adjust the eight accepted design patterns for sociality in human-robot interaction (Kahn et al., 2008) so that they reflect social and cultural norms in Africa. These design patterns include the initial introduction, didactic communication, moving in motion together, personal interests and history, recovering from mistakes, reciprocal turn taking, physical intimacy, and claiming unfair treatment or wrongful harms.

Having identified the verbal and non-verbal social and cultural norms of human interaction that are prevalent in different countries in Africa, i.e., cultural knowledge, we will encapsulate them in the modes of interaction of social robots so that these robots engage with African people in a manner that is consistent with their expectations of acceptable — courteous and respectful — social interaction, rather than using inappropriate or insensitive social behaviors and modes of interaction from the West or the East.

In pursuing this research, we recognize that there are many different cultures in Africa, with many different norms for deictic, iconic, and symbolic manual gesturing, as well as gestures involving eye gaze, head tilt, eyebrows, and body posture, generally. Similarly, there are many different ways in which spoken language can express nuances of meaning by modulating amplitude and timbre. In order to ensure that the project objectives can be achieved in the time available, we restrict the scope of the project to the cultures and social practices that are prevalent in Rwanda and South Africa.

The outcomes of the research will take the form of a suite of software primitives, integrated in an application programming system architecture, and a set of design patterns that can be recruited during human-robot interaction, deploying the spatial, non-verbal and verbal communication channels that are best suited to the social and cultural needs of the interaction.

The software primitives, system architecture, and design patterns will be evaluated in two complementary use cases. Each use case will be conducted in two phases, so that evaluation after the first phase can provide feedback and allow the results of the research to be adjusted and improved, if necessary.

1.4 Technical Scope

It is increasingly accepted that AI systems need to understand, and interact in, the social world of humans. This is particularly true in robotics, which is viewed by many as "cognition-enabled transferable embodied AI" (euROBIN, 2023) and, especially, in social robotics. As we noted above, effective interaction is essential for acceptance, trust, and adoption. This implies that social robots must be able to recognize cultural traits in humans, infer their intentions, and behave in a manner that is culturally legible and predictable by adhering to social and cultural norms.

A complete culturally competent robot requires at least five elements (Bruno et al., 2017a), as follows.

1. Cultural knowledge representation.
2. Culturally sensitive planning and action execution.
3. Culturally aware multimodal human-robot interaction.
4. Culture-aware human emotion recognition.
5. Culture identity assessment, habits, and preferences.

This research project focusses on the first three of these, i.e., the generation of culturally sensitive robot behavior. Specifically, the project does not address the challenge of learning these behaviors from observation, i.e., learning cultural knowledge through interaction. Instead, the

approach in this project is to catalogue the behaviors based on ethnographic research and embed them in reconfigurable software design patterns. In the short term, this is a more tractable approach and will produce reusable results, while still being compatible with the goal of developing culturally competent robots by combining top-down and bottom-up approaches based on the predetermined profiles of a cultural group and the cultural profiles derived from the behaviors of individuals, respectively (Khaliq et al., 2018).

Chapter 2

Work Plan

2.1 Approach and Activities

We adopt both a user-centric perspective and an agile and iterative approach in this project. This is reflected in the work plan; see the Pert chart in Figure 2.1 and the list of work packages in Table 2.1.

WP1 is user-driven and focusses on identifying the cultural and social norms, i.e., the cultural knowledge, that define respectful, engaging interaction in African countries. It will achieve this through ethnographic user studies that create the data that will then define the development of Africa-centric modes of human-robot interaction and design patterns for courteous and respectful sociality. The cultural knowledge and design patterns are used to specify the robot behaviors in WP5, which are used in the two demonstration and evaluation use cases in WP6. The robot sensing functionality required for interaction is developed in WP4. The integration of all functionality in a coherent system architecture is carried out in WP3, while the specification of the interaction scenarios is carried out in WP2. The interpretation of these scenarios is effected by the robot behaviors subsystem developed in WP5. Monitoring research progress, meanwhile, is also done in WP6 through user studies that test and validate the targeted use case functionality at the end of years 2 and 3 of the project, taking appropriate action after year 2 to adjust and augment each element in WP1 - WP5 in order to improve the performance in the use cases in the subsequent phase. The timeline also highlights this iterative development; see the Gantt chart in Figure 2 in Section 2.4, where detailed work package descriptions are also provided.

Table 2.1: List of work packages

WP No.	Work Package Title	Responsible Partner	Person Months	Start Month	End Month
1	African Cultural Knowledge		6.60	3	27
2	Interaction Scenario Specification		3.10	2	27
3	System Architecture & Systems Engineering		7.75	1	36
4	Robot Sensing		11.50	1	33
5	Robot Behaviors		9.00	1	33
6	Use Case Demonstration and Evaluation		4.50	19	36
7	Dissemination and Impact		6.00	1	36
8	Project Management		6.90	1	36

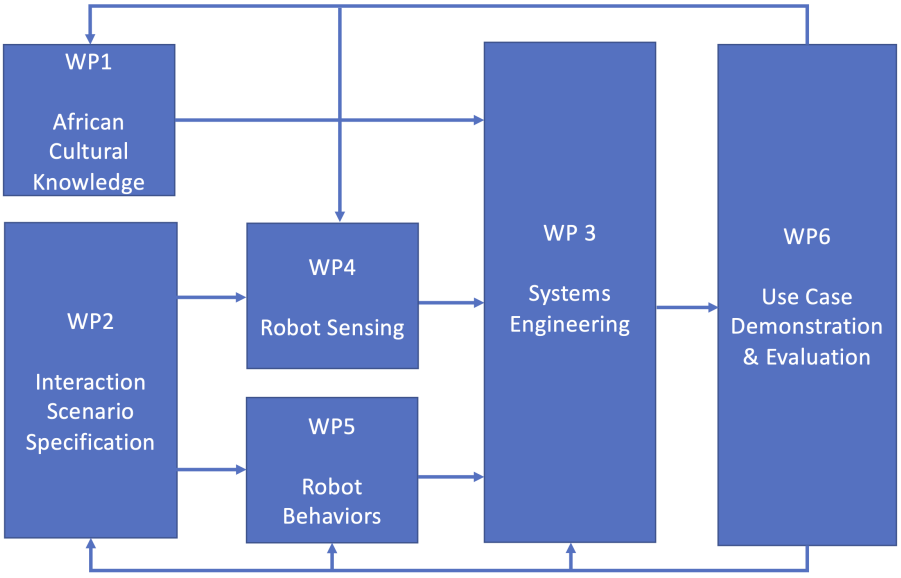

Figure 2.1: PERT chart showing the dependencies between the technical work packages; note the iterative development cycle.

For the ethnographic study of the cultural knowledge that defines respectful, engaging interaction in African countries that will be carried out in WP1, we perform these studies using two independent groups, and cross-validate the results, with one group validating the other group's results, adjusting appropriately, if necessary. In addition, we plan on engaging an external expert in ethnographic research in developing countries to ensure the validity of our approach and adapt it, as required.

In terms of technical development, we plan to adopt the development methodology and outline functional architecture for a culturally competent robot proposed by Bruno et al. (2017b), using the culture knowledge ontology proposed by Bruno et al. (2019) as a foundation.

From a software engineering perspective, CSSR4Africa will adopt the current best practice in robot software development based on component-based robotic engineering (Brugali & Scandurra, 2009; Brugali & Shakhimardanov, 2010) and adapted from established component-based software engineering of component-based software engineering (Heineman & Council, 2001; Szyperski, 2002). With a focus on effective integration, the CSSR4Africa system will adhere to the best practice of making components composable (the property that makes it easily integrated into a larger system) and systems compositional (the property to exhibit predictable performance and behavior if the performance and behavior of the components are known).

We will use ROS, a globally-used implementation of component-based software engineering (CBSE). Furthermore, we will adopt an integration-focussed approach to the development of the system architecture (Vernon et al., 2015) based on CBSE, in general, and the component-port-connector model (i.e., the publish and subscribe model), in particular. In essence, then, we propose an adaptive, compositional agent-based message-passing software architecture to bridge WP4 & WP5 functionality and WP6 use-case behaviors.

All software will adhere to a project-specific set of software specification, design, coding, and documentation standards. These standards will also be applied to every module or subsystem that comprises each of the two primary subsystems (robot sensing and robot behaviors). In this way, the complete CSSR4Africa system will be based on a modular decomposition with the same interface protocol and software engineering standards applying at every level of decomposition. This approach facilitates transparent configuration and incremental integration & test of the complete system, from component, to subsystem, to system as a whole.

This approach ensures that those responsible for these subsystems have sufficient freedom

to choose the design that suits the subsystem needs best while at the same time requiring them to adhere to project-wide standards of software engineering and quality assurance.

2.2 Expected Outcomes

The CSSR4Africa research project will produce four measurable outcomes, as follows.

1. The identification of the cultural factors that impact on the acceptance of social robots in Africa, the constraints they impose on spatial, non-verbal, and verbal communication and social behavior, and the preferred behavioral traits that are considered appropriate for human-robot interaction in Africa.

2. The development of a suite of culturally-sensitive robot interaction primitives derived from the cultural knowledge, implemented on a Pepper humanoid robot[1] These primitives will include, for example, maintenance of appropriate interpersonal distance, adjustment of head and gaze direction, deployment of arm movement and hand gestures, and adoption of body posture.

3. The creation of a set of design patterns for culturally-sensitive social interaction in human-robot interaction, tuned to the preferences of African people.

4. A demonstration of the effectiveness of these design patterns in two complementary use cases.

We will validate the research by developing a use case that involves a humanoid robot giving a guest or group of guests a tour of a typical university laboratory. For this, we will specify and implement the functional requirements of the tour (for example, what exhibits to show, what to say to explain their purpose, how to navigate from one exhibit to another) and then factor in the non-functional requirements that address the culturally sensitive interaction while executing the functional elements of the tour (for example, how to greet and address the guest, how to maintain their engagement, how to draw their attention to the exhibit, and how to lead them from one exhibit to another). A second use case involving the robot taking the role of a receptionist that can provide information and directions to visitors will also be developed.

As noted in Section 1.1, the potential benefits of these research outcomes include AI technology, i.e., social robotics, that is culturally sensitive and therefore much more likely to be accepted and adopted, contributing to the *inclusive* digital transformation of Africa while simultaneously building research capacity by providing a foundation for future research in human-robot interaction in the form of reusable software and by exposing students to the various aspects of successful research practice. Since the software will be made available on GitHub with an open source licence, it can be freely used by researchers and software developers in Africa. This will make a significant contribution to the capacity of these roboticists to develop their own bespoke culturally-sensitive social robot applications helping to create new software development jobs in social robotics. This will position African software developers to compete for their share of a large and quickly growing global market in social robotics.

2.3 General Project Information

Project Management Plan

Work package 8 is dedicated to project management, with separate tasks for project coordination and communication (8.1), administration (8.2), risk management (8.3), the creation of a consortium agreement (8.4), and the creation of a gender action plan (8.5). Details of the work involved in each task can be found in the work package descriptions below.

[1] https://www.aldebaran.com/en/pepper.

Facilities
We will use the Pepper social robot from Aldebaran, a part of the United Robotics Group, to develop and validate the culturally-sensitive spatial, non-verbal, and verbal behaviors.

Dissemination Strategy
Work package 7 is dedicated to dissemination and impact, with targetted tasks Online Presence (7.1), Dissemination Activities (7.2), Open-Source Software, Data, and Designs (7.3), and an end-of-project Summer School (7.4). Details of the work involved in each task can be found in the work package descriptions.

Measures of Success and Assessment
The evaluation of the success of the project is carried out in work package WP6, Task 6.2 Use Case Evaluation and Task 6.3 Use Case Re-Evaluation. The focus of this evaluation is to assess the degree to which African people rate the interaction experience with social robots equipped with the Africa-centric traits developed in this project higher than interaction experience with social robots equipped only with universal traits. Evaluation will explore the *holistic interaction experience*, i.e., the three perspectives on social interaction with robots in (Young et al., 2011). User studies will be used for evaluation in the first phase, while a combination of user, observational, and, if feasible, ethnographic studies in the second phase. Both between-subject and within-subject design will be used, with assessment based on the Robot Social Attribute Scale (RoSAS) (Carpinella et al., 2017).

Project Risks
Risks and risk mitigation plans are outlined in Table 4 below. Task 8.3 in WP8 Project Management is dedicated to risk management.

Project Work Plan
The project work plan is summarized in the Gantt chart in Figure 2. Detailed work package descriptions are provided below, along with a list of deliverables in Table 2, a list of milestones in Table 3, risk mitigation strategies in Table 4, and a summary of effort by partner in Table 5.

Collaboration
Effective collaboration between the PI, the Co-PI, and their research assistants will be achieved by several means, e.g., daily communication using the project Discord platform with channels for each task, weekly monitoring of progress, three-monthly in-person meetings, and six-monthly progress report, and risk management activities (see WP8, Tasks 8.1 and 8.3). A significant amount of the work required will be carried out in student projects, e.g., summer internship projects and student research projects. These will, where appropriate, be co-supervised by faculty from both CMU-Africa and Wits. Students will be encouraged to engage in collaborative software development on the robots at each site.

2.4 Work Package Descriptions

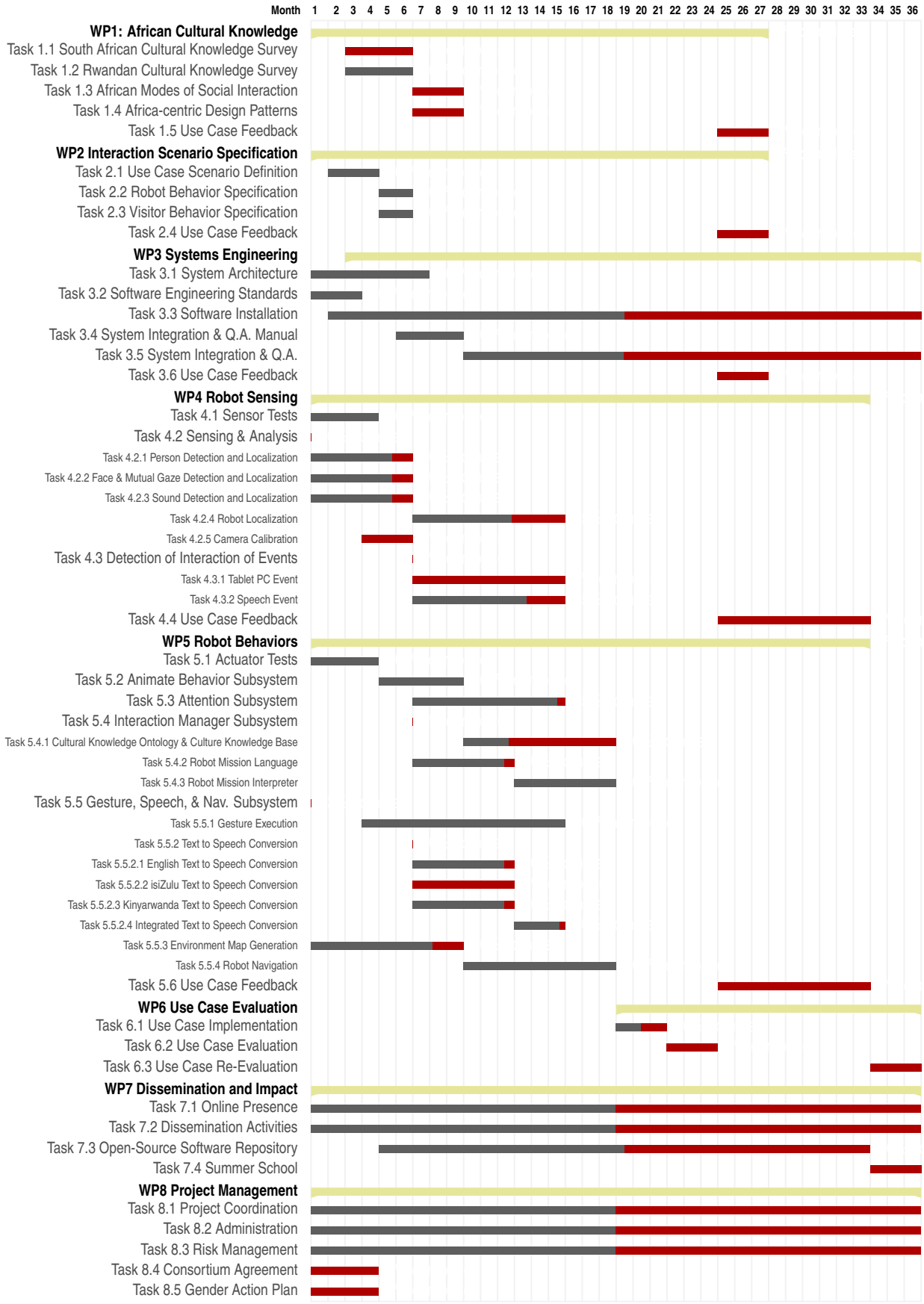

Figure 2: Gantt diagram showing the tasks in each work packages

2.4.1 Work Package 1: African Cultural Knowledge

Objectives

1. Identify the knowledge that is required to effect culturally-sensitive interaction in Rwanda and South Africa.
2. Use this cultural knowledge to identify, adjust, and augment the spatial, non-verbal, and verbal modes of interaction that impact the effectiveness of human-robot interaction (Bartneck et al., 2020).
3. Use this cultural knowledge to adapt and augment the eight design patterns for sociality in human-robot interaction proposed by Kahn et al. (2008).

Description of Work

Task 1.1: South African Cultural Knowledge Survey (M3–M6)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to carry out an ethnographic study with the goal of identifying the knowledge that is required to effect culturally-sensitive interaction in South Africa. This will form the basis of the general, population-based culture knowledge base for culturally sensitive human-robot interaction. This objective will be achieved by identifying the best practice for such ethnographic studies, preparing a plan of action to conduct the study, validating the plan with a qualified social scientist or ethnologist, adapting the plan as necessary, conducting the study, extracting the key results, summarizing the cultural knowledge, and validating it with a control group of people. The outcome of this task is described in Deliverable D1.1.

Task 1.2: Rwandan Cultural Knowledge Survey (M3–M6)
Responsible Partner: Carnegie Mellon University Africa.

The objective of this task is to carry out an ethnographic study with the goal of identifying the knowledge that is required to effect culturally-sensitive interaction in Rwanda. This will form the basis of the general, population-based culture knowledge base for culturally sensitive human-robot interaction. This objective will be achieved by identifying the best practice for such ethnographic studies, preparing a plan of action to conduct the study, validating the plan with a qualified social scientist or ethnologist, adapting the plan as necessary, conducting the study, extracting the key results, summarizing the cultural knowledge, and validating it with a control group of people. The outcome of this task is described in Deliverable D1.2.

Task 1.3: African Modes of Social Interaction (M7–M9)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to adjust or augment the spatial, non-verbal, and verbal modes of interaction that impact the effectiveness of human-robot interaction in Africa, focussing on Rwanda and South Africa, by incorporating cultural knowledge. This objective will be achieved by first cataloging the modes of interaction specified by Bartneck et al. (2020), analysing them to identify ways in which they should be adjusted to reflect cultural knowledge. A gap analysis will then be conducted to identify cultural knowledge that has not been taken into consideration. Any gaps will then be addressed by augmenting the existing modes of interaction. The task uses the cultural knowledge identified in Tasks 1.1 and 1.2, as documented in Deliverables D1.1 and D1.2. The outcome of this task is described in Deliverable D1.3.

Description of Work (continued)

Task 1.4: Africa-centric Design Patterns (M7–M9)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to adapt and augment the eight design patterns for sociality in human-robot interaction proposed by Kahn et al. (2008) by incorporating cultural knowledge. This objective will be achieved by analysing the eight design patterns to identify ways in which they should be adjusted to reflect cultural knowledge. A gap analysis will then be conducted to identify cultural knowledge that has not been taken into consideration. Any gaps will then be addressed by augmenting the existing design patterns. The task uses the cultural knowledge identified in Tasks 1.1 and 1.2, as documented in Deliverables D1.1 and D1.2. The outcome of this task is described in Deliverable D1.4.

Task 1.5: Use Case Feedback (M25–M27)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to update the South African and Rwandan cultural knowledge, the Africa-centric modes of interaction, and the Africa-centric design patterns for sociality in human-robot interaction based on feedback from the use case evaluations. This objective will be achieved by identifying any cultural knowledge that has not been included in the existing knowledge and identifying any cultural knowledge in the knowledge base that should be adjusted. The knowledge base will be updated, and the modes of interaction and design patterns will be amended accordingly. The task uses the outcomes of Task 6.2, as documented in Deliverable D6.2. The outcome of this task is described in revised versions of Deliverables D1.1, D1.2, D1.3, and D1.4.

List of Deliverables

Number	Deliverable Title	Responsible Partner	Delivery Month
D1.1	South African Cultural Knowledge, versions 1 and 2	Wits	M6, M27
D1.2	Rwandan Cultural Knowledge, versions 1 and 2	CMU-Africa	M6, M27
D1.3	African Modes of Social Interaction, versions 1 and 2	Wits	M9, M27
D1.4	Africa-centric Design Patterns, versions 1 and 2	Wits	M9, M27
D1.5	Updates to Deliverables D1.1, D1.2, and D1.3	Wits	M27

Description of Deliverables

D1.1 South African Cultural Knowledge
Deliverable type: report.
This deliverable represents the outcome of Task 1.1. It comprises a catalogue of general cultural knowledge[2] in the form of behaviors, activities, actions, and movements[3] that are either culturally sensitive or culturally insensitive. The culturally-sensitive behaviors, activities, actions, and motions will be used to define the culturally sensitive African modes of social interaction in Deliverable D1.3 and the Africa-centric design patterns in Deliverable D1.4. It will be formalized in the culture knowledge ontology and culture knowledge base in Deliverable D5.4.1.

D1.2 Rwandan Cultural Knowledge
Deliverable type: software and report.
This deliverable represents the outcome of Task 1.1. It comprises a catalogue of general cultural knowledge in the form of behaviors, activities, actions, and movements that are either culturally sensitive or culturally insensitive. The culturally-sensitive behaviors, activities, actions, and motions will be used to define the culturally sensitive African modes of social interaction in Deliverable D1.3 and the Africa-centric design patterns in Deliverable D1.4. It will be formalized in the culture knowledge ontology and culture knowledge base in Deliverable D5.4.1. In addition to the cultural knowledge, a dashboard will be developed for visualizing survey responses to aid with data cleaning and forming consensus views on each element of cultural knowledge. The dashboard software will be made freely available, and a user manual will be included in the deliverable report.

D1.3 African Modes of Social Interaction
Deliverable type: report.
This deliverable represents the outcome of Task 1.3. It comprises a catalogue of modes of social interaction that are culturally sensitive, based on the cultural knowledge contained in Deliverables D1.1 and D1.2. The modes of interaction include spatial interaction (proxemics, localization and navigation, socially appropriate positioning, initiation of interaction, communication of intent), nonverbal interaction (gaze and eye movement, deictic, iconic, symbolic, and beat gesture, mimicry and imitation, touch, posture and movement, and interaction rhythm and timing), and verbal interaction (speech, speech recognition, language understanding, speech generation) (Bartneck et al., 2020).

[2]We follow the lead of Bruno et al. (2017b) in distinguishing between general cultural knowledge, which is typically valid for many people in a given culture, and specific cultural knowledge, which is based on one or a small number of people.

[3]We define a behavior as a pattern of activity, an activity as a sequence of actions, an action as a goal-directed sequence of motions, and a motion as a change in effector pose, subject to constraints, e.g., trajectory or biological velocity and acceleration profile.

Description of Deliverables (continued)

D1.4 Africa-centric Design Patterns
Deliverable type: report.
This deliverable represents the outcome of Task 1.4. It describes any adjustments that need to be made to the eight accepted design patterns for sociality in human-robot interaction proposed by Kahn et al. (2008) so that they reflect social and cultural norms in Africa. The adjustment will be based on the cultural knowledge reported in Deliverable D1.1. These design patterns include the initial introduction, didactic communication, moving in motion together, personal interests and history, recovering from mistakes, reciprocal turn taking, physical intimacy, and claiming unfair treatment or wrongful harms. Additional design patterns will be added, if necessary, again based on the cultural knowledge contained in Deliverables D1.1 and D1.2.

D1.5 Updates to Deliverables D1.1, D1.2, D1.3, and D1.4
Deliverable type: report.
This deliverable represents the outcome of Task 1.5. Based on the evaluation of the two use cases in work package WP6, and the material in Deliverable D6.2 in particular, it sets out any changes or additions that need to be made to the South African cultural knowledge, Rwandan cultural knowledge, the Africa-centric modes of interaction, and the Africa-centric design patterns for sociality in human-robot interaction in Deliverables D1.1, D1.2, D1.3, and D1.4, respectively.

2.4.2 Work Package 2: Interaction Scenario Specification

Objectives

1. To develop the two use case scenarios — lab tour guide and receptionist — and define detailed procedures for their execution.
2. To specify in detail the desired robot behavior from the point of view of the robot's manager, i.e., the person responsible for assigning the role of lab tour guide or receptionist to the robot.
3. To specify in detail the expected visitor behavior, i.e., the expected interactions by the person to whom the tour is being given or by the person at reception. As noted already, this project does not address the interpretation of natural non-verbal human communication such as gestures and emotions. While limited automated speech recognition will be investigated, the main form of interaction by the human will be through the tablet PC on the robot.

Description of Work

Task 2.1: Use Case Scenario Definition (M2–M4)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to define the two use cases: lab tour guide and receptionist. This objective will be achieved by detailing the aim of the use case scenario, the setting, and the procedure: a step-by-step implementation of the scenario. The implementation will be based on a structured walk-through of all the interactions that instantiate the scenario. The purpose of the walk-through is to unwrap the interaction in each use case into micro-steps of elementary perceptions as seen by the robot (from the perspective of the robot's manager) and actions by the interaction partner, i.e., the visitor. Once this timeline of elemental perceptions and actions has been unwrapped, we can then identify the measurable sensory indicators required to parameterize and quantify the information about visitor that is necessary to allow the robot to interact effectively (i.e., in a culturally sensitive manner) with her or him, e.g., locating the position of the visitor, their face and gaze. As noted already, the main form of interaction by the human will be through the tablet PC on the robot and, optionally, through limited spoken requests and instructions, implemented with automated speech recognition. This unwrapped walk-through provides the baseline data for tasks T2.2 and T2.3. The outcome of this task is described in Deliverable D2.1.

Description of Work (continued)

Task 2.2: Robot Behavior Specification (M5–M6)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to generate a list of ROS nodes that will implement desired robot behaviors, i.e., sequences of actions, that the gesture, speech, and navigation subsystem in WP5 will later synthesize. This objective will be achieved by analysing the use case scenarios defined in Deliverable D2.1. The outcome of this task is documented in Deliverable 2.2.

Task 2.3: Visitor Behavior Specification (M5–M6)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to generate a list of ROS nodes that will implement the perceptual functions that will enable the robot to interact effectively with the visitor and interpret the visitor behaviors specified in the use case scenarios. This project does not address the interpretation of natural non-verbal human communication such as gestures and emotions. While limited automated speech recognition will be investigated, the main form of interaction by the human will be through the tablet PC on the robot. Nevertheless, the robot must be able to address the visitor, and this requires certain perceptual capabilities. Hence, the objective of this task is to identify the ROS nodes that will provide these capabilities. This objective will be achieved by analysing the use case scenarios defined in Deliverable D2.1. The outcome of this task is documented in Deliverable D2.3.

Task 2.4: Use Case Updates (M25–M27)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to update the use cases based on feedback from the use case evaluations. This objective will be achieved by identifying any adjustments that are necessary to improve the acceptability, feasibility, and cultural sensitivity of the use cases. The task uses the outcomes of Task 6.2, as documented in Deliverable D6.2, The outcome of this task is summarized in Deliverable D2.4 and incorporated in revised versions of Deliverables D2.1, D2.2, and D2.3.

List of Deliverables

Number	Deliverable Title	Responsible Partner	Delivery Month
D2.1	Use Case Scenario Definition	CMU-Africa	M4, M27
D2.2	Robot Behavior Specification	CMU-Africa	M6, M27
D2.3	Visitor Behavior Specification	CMU-Africa	M6, M27
D2.4	Use Case Updates	Wits	M27

Description of Deliverables

D2.1 Use Case Scenario Definition
Deliverable type: report.
This deliverable represents the outcome of Task 2.1. It presents a detailed scenario definition for the two use cases: lab tour guide and receptionist. It describes the aim of the use case, the setting, the procedure, and the measurable variables. It provides the basis for the robot behavior specification (D2.2) and the visitor behavior specification (D2.3). The report includes a walk-through of the scenario, providing a decomposition into a time sequence of elementary robot actions. For each action, the deliverable will specify the following.

1. The set of triggers for the action, e.g., input from the robot's manager or visitor using the tablet PC on the robot, speech input, or movements of the visitor.
2. The sensory cues that characterize each trigger, including tablet PC inputs.
3. The exact sequence of movements, expressions, or vocal output that constitute the robot action and their associated sensory cues.
4. The goal of the action, i.e., the expected change in the environment, the response of the visitor, or the robot's manager.
5. The sensory cues that characterize the goal of the action; in each case, there may be multiple triggers and responses.

The report will also detail the layout of the environment in which the scenarios are set.

D2.2 Robot Behavior Specification
Deliverable type: report.
This deliverable represents the outcome of Task 2.2. It identifies the robot's behaviour in the two use-case scenarios described in Deliverable D2.1 Use Case Scenario Definition, Sections 3 and 4 of which define the robot actions, component movements and sensory cues, and the associated sensory-motor process specified in the use case interactions.
The purpose of this deliverable is simply to identify the ROS nodes that will implement these actions. It also identifies the ROS packages in which the ROS nodes will be implemented. The detailed specification of these nodes will be provided in Deliverable D3.1 System Architecture. The dynamics of the interaction between the visitor and the robot will be specified in Deliverable D5.4.2 Robot Mission Language, and implemented in Deliverable D5.4.3 Robot Mission Interpreter.

Description of Deliverables (continued)

D2.3 Visitor Behavior Specification
Deliverable type: report.
This deliverable represents the outcome of Task 2.3. It identifies the visitor's behaviour in the two use-case scenarios described in Deliverable D2.1 Use Case Scenario Definition, Sections 3 and 4 of which define the visitor actions and describe them in perceptual terms, from the perspective of the robot.
The purpose of this deliverable is simply to identify the ROS nodes that will provide this sensory functionality. It also identifies the ROS packages in which the ROS nodes will be implemented. The detailed specification of these nodes will be provided in Deliverable D3.1 System Architecture.
The dynamics of the interaction between the visitor and the robot will be specified in Deliverable D5.4.2 Robot Mission Language, and implemented in Deliverable D5.4.3 Robot Mission Interpreter.

D2.4 Use Case Updates
Deliverable type: report.
This deliverable represents the outcome of Task 2.4. Based on the evaluation of the two use cases in work package WP6, and the material Deliverable D6.2 in particular, it sets out any changes or additions that need to be made to the use case scenario definition, the robot behavior specification, and the visitor behavior specification in deliverables D2.1, D2.2, and D2.3, respectively.

2.4.3 Work Package 3: Systems Engineering

Objectives

1. Design a software architecture that will facilitate the integration of the results of the project into a complete operational culturally sensitive social robotics system.
2. Formalize the integration process by identifying appropriate software engineering standards and quality assurance procedures.
3. Carry out this integration and quality assurance process.

Description of Work

Task 3.1: System Architecture (M1–M7)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to design the system architecture to integrate the functionality developed in work packages WP4 and WP5, providing the necessary infrastructure to implement each of the two use cases, including domain- and task-specific knowledge. This objective will be achieved by identifying each of the architecture subsystems, specifying their functionality, and defining their interfaces using a component-based software engineering (CBSE) approach (Vernon et al., 2015). We plan to adopt the development methodology and outline functional architecture for a culturally competent robot proposed by Bruno et al. (2017b), using the ontology proposed by Bruno et al. (2019) as a foundation. The task is guided by the requirements encapsulated in Deliverables D2.1 User Case Scenario Definition, D2.2 Robot Behavior Specification, and D2.3 Visitor Behavior Specification. Architecturally, the software developed in work packages WP4 and WP5 will be viewed as a modular subsystems. The goal of this task is to specify the functionality of each of these subsystems and to define their interfaces, i.e., the manner in which data and control signals are input to and output from these subsystems. These interfaces will be specified at both a high level of abstraction (e.g., what data is input and output and how it is represented) and at a low-level of abstraction (how this data is exposed through subsystems interfaces and how the functionality of the subsystem can be externally configured by other subsystems). The low-level interface and functional configuration facilities will be effected using the ROS (Robot Operating System). Interface design and module specification will be done in collaboration with the respective module development tasks in work packages WP4 and WP5. The outcome of this task is described in Deliverable D3.1.

Description of Work (continued)

Task 3.2: Software Engineering Standards (M1–M3)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to define a set of project standards governing the specification, design, documentation, and test of all software to be developed in work packages WP4 and WP5. The specification standards will address functional definition, data representation, and module / sub-system behavior. Design standards will focus on the decoupling of functional computation, module communication, external module configuration, and inter-module coordination. To maximize compositionality, a best-practice component-based software engineering model. To enhance dependability, software associated with the normal flow of execution and exception handling will be separated. Software test strategies will include black-box unit testing, white-box structural testing at the sub-system level, and regression testing to ensure backward compatibility. Acceptance tests will be carried out on the basis of the required behaviors for the two use cases. In addition, this task will define the procedures whereby software developed in work packages WP4 and WP5 is submitted for integration, checked against standards, and tested. These procedures will also address the actions to be taken as a result of this integration and quality assurance process. This objective will be achieved by adapting and adopting standards used in previous research projects RobotCub[4] and DREAM.[5] The outcome of this task is described in Deliverable D3.2.

Task 3.3: Software Installation (M2–M36)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to document the process for the installation and execution of the software required to instantiate all or part of the CSSR4Africa system and run the unit, integration, and system tests, and the two case scenarios on the physical robot. This objective will be achieved by documenting the installation of all software components on an ongoing basis, as a living document, to ensure it reflects the current capabilities of the system. The outcome of this task is described in Deliverable D3.3. Official versions will be produced every six months.

Task 3.4: System Integration and Quality Assurance Manual (M6–M9)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to write a manual specifying the procedures to be followed when developing software in the project, submitting it for integration into the current system release, the acceptance unit tests, and the follow-up action required if a module does not pass an acceptance unit test. It identifies the processes that are used to ensure that all software included in the system release adheres to the standards set out in Deliverable D3.2 and follows the related guidelines. This objective will be achieved by adapting and adopting standards used in previous research projects RobotCub and DREAM. The outcome of this task is described in Deliverable D3.4.

[4] http://www.robotcub.org/
[5] https://dream2020.github.io/DREAM/

Description of Work (continued)

Task 3.5: System Integration and Quality Assurance (M10–M36)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to incorporate software submitted for integration in the current version of the CSSR4Africa system. This objective will be achieved by running the accompanying unit tests and subjecting the submitted software to integration tests. If the software passes these test, it will then be integrated into the current version. In this way, the software is subjected to strict quality assured. This process will be carried out both on demand, as software is submitted for integration by those responsible for the work in work packages WP4 and WP5, and as required by check-points defined by the project milestones. The task uses the outcome of Task 3.4 as documented in Deliverable D3.4. The outcome of this task is described in Deliverable D3.5.

Task 3.6: Use Case Feedback (M25–M27)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to update the system architecture based on feedback from the use case evaluations. This objective will be achieved by identifying any adjustments that are necessary to improve the acceptability, performance, and cultural sensitivity of the use cases. The task uses the outcomes of Task 6.2, as documented in Deliverable D6.2, The outcome of this task is summarized in Deliverable D3.6 and incorporated in revised versions of Deliverables D3.1.

List of Deliverables

Number	Deliverable Title	Responsible Partner	Delivery Month
D3.1	System Architecture	CMU-Africa	M7, M30
D3.2	Software Engineering Standards Manual	CMU-Africa	M3
D3.3	Software Installation Manual	CMU-Africa	M6, ..., M36
D3.4	System Integration & Quality Assurance Manual	CMU-Africa	M9
D3.5	System Integration and Quality Assurance	CMU-Africa	M12, M24, M36
D3.6	Use Case Feedback	CMU-Africa	M27

Description of Deliverables

D3.1 System Architecture
Deliverable type: report.
This deliverable represents the outcome of Task 3.1. It specifies the CSSR4Africa system architecture in detail, identifying the component subsystems, the modules comprising each subsystem, and the information exchanged between subsystems and modules, including a specification of the data that are input to each module, the data that are output from each module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through ROS topics, services, or actions. Specifically, the deliverable provides the functional specification of the two subsystems to be developed in work package WP4 Robot Sensing (Sensing and Analysis, and Detection of Interaction Events) and the four subsystems to be developed in work package WP5 Robot behaviors (Animate Behavior, Attention, Interaction Manager, and Gesture, Speech, & Navigation). All data will be specified by both information content and representation.

D3.2 Software Engineering Standards Manual
Deliverable type: report.
This deliverable represents the outcome of Task 3.2. It is a manual of software engineering standards comprising five parts, as follows.

1. Module and subsystem specification addressing, e.g., requirements model & use cases, computational model, functional model & functional decomposition, data model, process flow models, behavioral model, class and class-hierarchy definition, object-relationship model, object-behavior model.

2. Module design addressing, e.g., data representation and data-structure definition, algorithm realization (calling sequence, API, method invocation), messaging protocols, control mechanisms (task management, HCI management).

3. Implementation, addressing coding guidelines and internal documentation (comment) guidelines.

4. Testing procedures, addressing, e.g., unit and regression testing, backward compatibility.

5. Documentation, specifically external documentation such as reference manuals, user manuals, configuration, and test procedures.

Description of Deliverables (continued)

D3.3 Software Installation Manual
Deliverable type: report.
This deliverable represents the outcome of Task 3.3. It documents the process for the installation and execution of the software required to instantiate all or part of the CSSR4Africa system and run the unit, integration, and system tests, and the two case scenarios on the physical robot. The deliverable will be created as a living document to ensure it reflects the current capabilities of the system. Official versions will be produced every six months.

D3.4 System Integration and Quality Assurance Manual
Deliverable type: report.
This deliverable represents the outcome of Task 3.4. It is a manual specifying the procedures to be followed when developing software in the project, submitting it for integration into the current system release, the acceptance unit tests, and the follow-up action required if a module does not pass an acceptance unit test. It identifies the processes that are used to ensure that all software included in the system release adheres to the standards set out in Deliverable D3.2 and follows the related guidelines.

D3.5 System Integration and Quality Assurance
Deliverable type: software, report.
This deliverable represents the outcome of Task 3.5. It comprises the functional integrated software at different stages of the system development. In addition to functional code, the deliverable will include a report on the results of the integration tests. The deliverable will be issued annually over the duration of the project. The final version will present the final system test and a test report, along with a system user manual and a system reference manual.

D3.6 Use Case Feedback
Deliverable type: report.
This deliverable represents the outcome of Task 3.6. It documents any adjustments that need to be made to the system architecture based on the evaluation of the two use cases in work package WP6.

2.4.4 Work Package 4: Robot Sensing

Objectives

1. Develop a suite of unit tests to verify that all sensors are functioning correctly and that the sensor data can be accessed using ROS.
2. Identify and implement algorithms that will detect and localize people, including their face, localize sounds, and localize the robot in a world frame of reference.
3. Detect interaction events on the robot's tablet PC.
4. Detect speech events and perform automated speech recognition.

Description of Work

Task 4.1: Sensor Tests (M1–M4)
Responsible Partner: Carnegie Mellon University Africa.

The objective of this task is to write a suite of unit tests, one for each sensor, to verify that sensor data is successfully acquired on each sensor topic. A subset of the unit tests can be selected, and those that are selected can be run in sequence or in parallel.

This objective will be achieved by creating a list of sensors and the associated topics used by the physical Pepper robot and the Pepper robot simulator, writing a suite of unit tests that will verify the operation of each sensor on both the physical Pepper and the simulator, writing a unit test scheduling program that will use an external configuration file to identify the tests to be run, the manner in which they are to be run (in sequence or in parallel), and the platform on which they are to be run (the physical robot or the simulator). The software development process for each unit test and the scheduling program will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are detailed in Deliverable D4.1.

Task 4.2: Sensing & Analysis (M1–M15)
Responsible Partner: Carnegie Mellon University Africa.
The sensing and analysis task is concerned with analyzing sensor data to detect and localize interaction partners with respect to the robot, and localize the robot with respect to its environment. Combined with the cultural knowledge, this information will enable the robot to pay attention to people and interact with them in a culturally sensitive manner by respecting social and cultural norms for proxemics and eye contact. It also supports robot navigation and deictic gestures by the robot to objects in the robot's environment. There are four subtasks. Three focus on the interaction partners and are concerned with detecting and localizing (i) people, (ii) faces, (iii) sounds, e.g., voices. A fourth subtask is concerned with localizing the robot in the world frame of reference. There is a deliverable for each of the subtasks, rather than a single deliverable for the task overall.

Description of Work (continued)

Task 4.2.1: Person Detection and Localization (M1–M6)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to develop a software module to detect and localize people in the robot's field of view. If a person is detected, her or his position will be determined both in an image frame of reference and in the robot's Cartesian head frame of reference. If needed, the location of the person in a world frame of reference can be computed based on the pose of the robot head with respect to the robot base (determined from the robot kinematic model and the knee pitch, hip pitch, hip roll, head pitch, and head yaw angles) and the pose of the robot base in the world frame of reference (determined in Task 4.2.4). Knowledge of the position of the detected person in either the robot's base frame of reference or the world frame of reference is needed if the robot is required to navigate with respect to the person, e.g., when approaching them or maintaining a respectful distance from them.

Note that this module computes the person's position only, not their pose, i.e., position and orientation. In addition, the region that the person occupies in the image will also be determined by computing the bounding box surrounding the person. This bounding box will be drawn on an output image. If more than one person is present in the robot's field of view, all of them should be detected and localized.

To ensure coherence in detection and localization over time, each detected person should be labelled in the image (e.g., "Person 1") and the same label should be assigned to that person in subsequent images. The label and the bounding box should be colour-coded, assigning different colours to different people, and the same colour is to be given person in each image in a sequence of images. If that person is no longer detected in an image, then that label should not be reused. If that person reappears in a subsequent image, she or he will be given a new label. As such, this module is only concerned with consistent detection of people over time, not recognition of previously detected people. It is assumed that people don't change between images. Consequently, a person in one image is deemed to be the same one in a previous image if the spatial displacement of the person is less than a given tolerance (to specified by a configuration parameter value). Each detection is to be assigned a confidence value between 0 and 1 indicating the likelihood that the detection is not a false positive. The camera to be used for image acquisition and the depth sensor to be used for 3D localization will be specified in a module configuration file..

The task objective will be achieved by first conducting a review of person detection and localization algorithms, focussing on standard approaches. Three candidate approaches, such as Histogram of Oriented Gradients (HOG), Region-based Convolutional Neural Networks (R-CNN), and deep learning-based models such as YOLO (You Look Only Once), will be implemented for comparison, the best of which (in terms of minimizing false reject and false accept errors) will be selected for use in the project.

Description of Work (continued)

For localization in the robot's Cartesian head frame of reference, the depth image provided by the RGB-D camera depth sensor will be fused with the visual image (in the case that the visual image is acquired with the forward looking camera, rather than the RGB-D camera). After camera calibration (performed in Task 4.2.5), the camera model will be used with the inverse perspective transformation, along with the depth image, to compute the location of the person in the robot's head frame of reference, and, using the robot kinematic model and the knee pitch, hip pitch, hip roll, and head pitch, and head yaw joint angles, in the robot's base frame of reference. The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The outputs of each of these phases are detailed in Deliverable D4.2.1.

Note that this module does not support the simulator because it can optionally use an external RealSense camera which is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Work (continued)

Task 4.2.2: Face Mutual Gaze Detection and Localization (M1–M6)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to develop a software module to detect and localize the faces and determine the direction of gaze of people in the robot's field of view. If a face is detected, its position will be determined by both an image frame of reference and the robot's Cartesian head frame of reference. In addition, the region that the face occupies in the image will also be determined by computing the bounding box surrounding the face. This bounding box will be drawn on an output image. If more than one face is present in the robot's field of view, then all of them should be detected and localized.

To ensure coherence in detection and localization over time, each detected face should be labelled (e.g., "Face 1") and the same label should be assigned to that face in subsequent images. The label and the bounding box should be colour-coded, assigning different colours to different people, and the same colour to a given person in each image in a sequence of images. If that face is no longer detected in one or more images (the number to be specified in a configuration parameter value) due to, for example, a false reject error, then that label should not be reused. If that face reappears in a subsequent image, it will be given a new label. As such, this module is only concerned with consistent detection of faces over time, not recognition of previously detected faces, and it is assumed that faces don't change between images. Consequently, a face in one image is deemed to be the same one in a previous image if the spatial displacement of the face is less than a given tolerance (specified by a configuration parameter value). Each detection is to be assigned a confidence value between 0 and 1 indicating the likelihood that the detection is not a false positive. The camera to be used for image acquisition will be specified in a module configuration file.

This objective will be achieved by utilizing Mediapipe and deep learning models such as YOLO (You Look Only Once) for detection and a simple centroid tracker algorithm for tracking. The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The outputs of each of these phases are detailed in Deliverable D4.2.2.

For each detected and localized face, this task will determine whether the person is gazing directly at the robot. This provides a simplified approach to assessing mutual gaze between the robot and a human. The system will leverage pre-trained deep learning models for head pose estimation to achieve this functionality.

Note that this module does not support the simulator because it can optionally use an external RealSense camera which is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Work (continued)

Task 4.2.3: Sound Detection and Localization (M1–M6)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to develop a software module to detect and identify the direction of arrival of a conspicuous sound within the robot's hearing range. The direction will be limited to the azimuth (i.e., horizontal) plane. This will allow the robot to direct its gaze in the direction of the sound. If a sound is detected, its angle of arrival will be determined in the robot's Cartesian head frame of reference. The audio signal of the detected sound, from onset of the sound to offset of the sound, will be captured for output.

This objective will be achieved by computing the interaural time delay (ITD) between the arrival of the sound at the robot's microphones on the top of the robot's head. Other techniques will also be investigated. The module will be tuned to detect human voices rather than ambient sounds or background noise by using signal processing techniques such as band-pass filtering and noise reduction. The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The outputs of each of these phases are detailed in Deliverable D4.2.3.

Note that this module does not support the simulator because the microphones are not available in Gazebo; see the notes on Version 3.0 in the Document History.

Task 4.2.4: Robot Localization (M7–M15)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to develop a software module to determine the pose (position and orientation) of the robot in a Cartesian world frame of reference.

This objective will be achieved by a combination of relative position estimation and absolute position estimation. For relative position estimation, both odometry and the use of the robot's inertial management unit will be investigated. For absolute position estimation, we will investigate triangulation using three landmarks. This will require the development of functionality for landmark recognition. For this, we will investigate the use of SIFT (Scale Invariant Feature Transform) and YOLO (You Look Only Once). This task also requires the existence of a map of the environment. For this, the output of Task 5.5.3 Environment Map Generation, encapsulated in Deliverable D5.5.3, will be used. To estimate the orientation of the robot, we will ignore adjustments of body posture through rotation about the X- and Y-axes, and we will restrict estimation to rotation about the robot's Z-axis in the based frame of reference, i.e., rotation about the Z-axis. This rotation angle can be recovered by determining the direction given by the line of sight from the robot to one of the landmarks, and adjusting for any rotation about the Z-axis of the robot's head frame of reference with respect to the base frame of reference. The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The outputs of each of these phases are detailed in Deliverable D4.2.1. The performance of the module will be compared to the localization performance of the SLAM (Simultaneous Localization and Mapping) module that will be developed in Task 5.5.3.

Note that this module does not support the simulator because it can optionally use an external RealSense camera which is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Work (continued)

Task 4.2.5: Camera Calibration (M4–M6)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to develop a software module to determine the 3×4 camera model matrix for the robot's front camera and stereo cameras. The camera model parameters can then be used to compute the inverse perspective transformation which, in turn, can be used to compute the 3D coordinates of an image point, provided the distance to that point is known (e.g., using the robot's depth sensor). Alternatively, one can use two camera models for the stereo cameras, and two inverse perspective transformations, to compute the 3D coordinates of an image point.

This objective will be achieved by extracting the 2D image control point coordinates from two or more images of a standard checkerboard calibration grid, computing the camera model based on these control points and the corresponding 3D world coordinates. The 3D coordinates of the calibration grid will be determined empirically, by placing it at a known position and orientation with respect to the robot head, and computing the coordinates from the spacing of the grid pattern and the grid pose. Thus, the 3D coordinates that are computed using the inverse perspective transformation are defined with respect to the robot head frame of reference.

Knowing the pose of the robot base with respect to the world frame of reference from the robot localization module (Deliverable D4.2.4), and knowing the pose of the robot head with respect to the robot base from the robot kinematic model and the knee pitch, hip pitch, hip roll, head pitch, and head yaw angles, the coordinates of this point the world frame of reference can be determined.

Note that this module does not support the simulator because it can optionally use an external RealSense camera which is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Work (continued)

Task 4.3: Detection of Interaction Events (M7–M15)
Responsible Partner: University of the Witwatersrand.
Detection of interaction events is a fundamental task that enables robots to perceive and interpret actions exhibited by humans. This capability empowers robots to understand and respond appropriately to their environment and interact with people. Since the development of capabilities for such natural interaction would require much more effort than is feasible in a project the size of CSSR4Africa, we restrict interaction events to simple requests using the tablet PC on the robot (Task 4.3.1). The exact functionality of events depends on the scenario requirements that will be identified in Task 2.1 Use Case Scenario Definition and, in particular, in Task 2.3 Visitor Behavior Specification. In anticipation of more natural interaction in the future, we include an additional task to address automated speech recognition for Kinyarwanda speakers.

Task 4.3.1: Tablet PC Event (M7–M15)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to develop a software module that that provides a ROS service to display a menu of interaction options that are appropriate in the current context of the scenario, wait for the visitor to select an option, and return the option selected.

This objective will be achieved by developing an Android touch screen menu display and option selection application. The menu of options will be driven by the requirements documented in Deliverable D2.3 Visitor Behavior Specification. The selected option will be used in the interaction manager subsystem to trigger the appropriate response by the robot. The module software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are detailed in Deliverable D4.3.1.

Note that this module does not support the simulator because the tablet PC is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Task 4.3.2: Speech Event (M7–M15)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to train, test, and deploy a speech-to-text model on the Pepper robot for detecting utterances made by an interaction partner in Kinyarwanda or English, and transcribing them into written text.

This objective will be achieved by analysing an audio signal of a sound detected by the module developed in Task 4.2.3 Sound Detection and Localization. Specifically, it will input the audio signal into deep neural networks that have been trained on Kinyarwanda and English data sets. The output of this network is the transcribed text. The module software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are detailed in Deliverable D4.3.2.

Note that this module does not support the simulator because it is dependent on input from sound detection and localization which does not support the simulator; see the notes on Version 3.0 in the Document History.

Description of Work (continued)

Task 4.4: Use Case Feedback (M25–M33)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to update the robot sensing based on feedback from the use case evaluations. This objective will be achieved by identifying any adjustments that are necessary to improve the acceptability, performance, and cultural sensitivity of the use cases. The task uses the outcomes of Task 6.2, as documented in Deliverable D6.2, The outcome of this task is summarized in Deliverable D4.4 and incorporated in revised versions of Deliverables D4.2.1 – D4.2.5, and D4.3.1 – D4.3.2.

List of Deliverables

Number	Deliverable Title	Responsible Partner	Delivery Month
D4.1	Sensor Tests	CMU-Africa	M4
D4.2.1	Person Detection and Localization	CMU-Africa	M6
D4.2.2	Face & Mutual Gaze Detection and Localization	CMU-Africa	M6
D4.2.3	Sound Detection and Localization	CMU-Africa	M6
D4.2.4	Robot Localization	CMU-Africa	M15
D4.2.5	Camera Calibration	Wits	M6
D4.3.1	Tablet PC Event	Wits	M15
D4.3.2	Speech Event	CMU-Africa	M15
D4.4	Use Case Feedback	CMU-Africa	M33

Description of Deliverables

D4.1 Sensor Tests
Deliverable type: software and report
This deliverable represents the outcome of Task 4.1. It comprises the documented software required to build and launch a module implemented as a single ROS node that encapsulates a suite of unit tests to verify that sensor data is successfully acquired on each sensor topic. The ROS node will be named sensorTest.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in the task description, identifying gaps and misalignments with actual needs.

The module specification section will define the functional characteristics, as follows. The sensors to be tested are to be identified by reading an associated input file that contains a list of key-value pairs. The keys will be the sensor names; these will be the same as those used in the Pepper documentation. The values will be either True or False, e.g., BumperA True. The sensor will be tested if the value is True. The keys and key values are not case sensitive. The input data file will be named sensorTestInput.dat.

The names of the topics to be used for each sensor will be read from a data file comprising a sequence of key-value pairs. The key is the name of the sensor. The value is the topic name. There will be two data files, one for the physical robot and another for the simulator.

The operation of the unit test module is determined by the contents of a configuration file that contains a list of key-value pairs. One key-value pair will specify the platform on which the tests are to be run, i.e., the physical Pepper robot or the Pepper simulator (e.g., platform robot | platform simulator). One key-value pair will specify the mode in which the unit tests should be run: in sequence or in parallel (e.g., mode sequential | mode parallel). One key-value pair will specify the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). One key-value pair will specify the filename of the file in which the simulator sensor and actuator topic names are stored (e.g., simulatorTopics simulatorTopics.dat). The configuration file will be named sensorTestConfiguration.ini.

Description of Deliverables (continued)

The results of each test should be written to an output file. Tests should also show the data acquired from the sensor in an appropriate manner, e.g., by displaying the images acquired from a camera or playing the sounds on a loudspeaker. The output file will be named sensorTestOutput.dat.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through files, or ROS subscribers, services, or actions. Based on the module specification in the task description, it is planned that the interface of this module will use both the physical robot and the simulator as drivers to generate test data.

The module design section will specify appropriate data structures that will be used to represent and store the acquired sensor data.

The coding section will contain the functional program code with internal documentation, any required build files, and driver program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will document the results of these using this sensorTest module.

A user manual will be included in the deliverable, explaining how to invoke, provide input to, and configure the sensorTest module.

Description of Deliverables (continued)

D4.2.1 Person Detection and Localization
Deliverable type: software and report
This deliverable represents the outcome of Task 4.2.1. It comprises the documented software required to build and launch a module implemented as a single ROS node to detect and determine the location of people in Pepper's field of view. The ROS node will be named personDetection.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in the task description, identifying gaps and misalignments with actual needs.

The module specification section will define the functional characteristics, as follows. The module will detect the presence of people in the robot's field of view and compute their position in an image frame of reference and in the robot's head Cartesian frame of reference. In addition, the region that the person occupies in the image will also be determined by computing the bounding box surrounding the person. If more than one person is present in the robot's field of view, then all of them should be detected and localized. To ensure coherence in detection and localization over time, each detected person should be labelled (e.g., "Person 1") and the same label should be assigned to that person in subsequent images. The label and the bounding box should be colour-coded, assigning different colours to different people, and the same colour to a given person in each image in a sequence of images. If that person is no longer detected in one or more images (the number to be specified in a configuration parameter value) due to, for example, a false reject error, then that label should not be reused. If that person reappears in a subsequent image, she or he will be given a new label. As such, this module is only concerned with consistent detection of people over time, not recognition of previously detected people, and it is assumed that people don't change between images. Consequently, a person in one image is deemed to be the same one in a previous image if the spatial displacement of the person is less than a given tolerance (to specified by a configuration parameter value).

The input will take the form of an RGB image from one of the robot's cameras and a depth image from one of the robot's depth sensors.

The output will take the form of an RGB image and a depth image, with bounding boxes drawn around each detected person, and an array of records, one record for each person detected. The components of a record are the person label, the 2D image coordinates denoting the centroid of the bounding box, the width and height of the bounding box, a confidence value between 0 and 1 indicating the likelihood that the detection is not a false positive, and the 3D coordinates that define the point that corresponds to the centroid of the bounding box surrounding the person in the image. The array of records will be published on a topic named /personDetection/data. The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal. Also, if the module is operating in verbose mode, the RGB image and the depth image will be displayed in an OpenCV window.

Description of Deliverables (continued)

The names of the topics to be used for each sensor will be read from a data file comprising a sequence of key-value pairs. The key is the name of the sensor. The value is the topic name.

The operation of the module will be determined by parameters provided in a configuration file that contains a list of key-value pairs. One key-value pair will specify the RGB camera to be used (e.g., camera FrontCamera | camera RealSense). One key-value pair will specify the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). One key-value pair will specify the number of images in which a person is not detected before it will be assigned a new label if and when it reappears at the same location (e.g., falseRejectTolerance 1 indicates that if the person is not detected in one image but reappears at that location in the next, it will be assigned the same label). One key-value pair will specify the spatial tolerance, given as a percentage of the width of the image, for a person to be assigned the same label (e.g., spatialTolerance 2). One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true). One key-value pair will specify the algorithm to use (e.g., algorithm HOG | YOLO). The configuration file will be named personDetectionConfiguration.json.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through ROS topics, services, or actions. In addition, it will include the specification of a driver to source test input data, and a specification of stub to sink test output data. Based on the module specification, it is planned that the interface of this module will use the physical robot as the primary driver to generate test data. Furthermore, it is planned to develop a driver to generate test data by acquiring RGB images from a USB camera or a USB RGB-D camera. This driver will publish the data on the same topics as those used by the physical robot. It is planned to develop an stub module to subscribe to the /personDetection/data topic and write the 2D location data to the screen.

The module design section will specify selected algorithms and data structures, and develop a unit test plan. It is planned to use deep learning models such as YOLO to detect and localize person within the robot's field of view.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

Description of Deliverables (continued)

The unit testing section will present the unit tests to verify that the module meets the module specification under different conditions, e.g., partial occlusion of the person and different lighting conditions. A unit test will be provided for each of the two test platforms: driver-stub unit test and physical robot unit test. The unit tests will be invoked by a launch file that brings up the required resources, e.g., any software to communicate the physical robot, the simulator, or the driver and stub. The driver will publish the sensor data on the physical robot sensor topics. A test report will document the results of these tests.

In addition, a user manual will be included in the deliverable.

Note that this module does not support the simulator because it can optionally use an external RealSense camera which is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D4.2.2 Face & Mutual Gaze Detection and Localization
Deliverable type: software and report
This deliverable represents the outcome of Task 4.2.2. It comprises the documented software required to build and launch a module implemented as a single ROS node to detect and determine the location of faces in Pepper's field of view and their respective gaze direction. The ROS node will be named faceDetection.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in the task description, identifying gaps and misalignments with actual needs.

The module specification section will define the functional characteristics, as follows. The module will detect the presence of faces of people in the robot's field of view and compute their position, and the direction of the gaze, in an image frame of reference. In addition, the region that the face occupies in the image will also be determined by computing the bounding box surrounding the face. If more than one face is present in the robot's field of view, then all of them should be detected and localized. To ensure coherence in detection and localization over time, each face should be labelled (e.g., "Face 1") and the same label should be assigned to that face in subsequent images. The label and the bounding box should be colour-coded, assigning different colours to different people, and the same colour to a given person in each image in a sequence of images. If that face is no longer detected in one or more images (the number to be specified in a configuration parameter value) due to, for example, a false reject error, then that label should not be reused. If that face reappears in a subsequent image, it will be given a new label. As such, this module is only concerned with consistent detection of faces over time, not recognition of previously detected faces, and it is assumed that faces don't change between images. Consequently, a face in one image is deemed to be the same one in a previous image if the spatial displacement of the face is less than a given tolerance (to specified by a configuration parameter value).

The input will take the form of an RGB image from one of the robot's cameras, and a depth image from one of the robot's depth sensors.

The output will take the form of an RGB image, with bounding boxes drawn around each detected face and an array of records, one record for each face detected. The components of a record are the face label, the 2D image coordinates denoting the centroid of the face bounding box, the width and height of the bounding box, a confidence value between 0 and 1 indicating the likelihood that the detection is not a false positive, and the 2D image coordinates of each eye. The array of records will be published on a topic named /faceDetection/data. The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal. Also, if the module is operating in verbose mode, the RGB image will be displayed in an OpenCV window.

Description of Deliverables (continued)

The names of the topics to be used for each sensor will be read from a data file comprising a sequence of key-value pairs. The key is the name of the sensor. The value is the topic name.

The operation of the module will be determined by parameters provided in a configuration file that contains a list of key-value pairs. One key-value pair will specify the camera to be used (e.g., camera FrontCamera | camera RealSense). One key-value pair will specify the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). One key-value pair will specify the number of images in which a face is not detected before it will be assigned a new label if and when it reappears at the same location (e.g., falseRejectTolerance 1 indicates that if the face is not detected in one image but reappears at that location in the next, it will be assigned the same label). One key-value pair will specify the spatial tolerance, given as a percentage of the width of the image, for a face to be assigned the same label (e.g., spatialTolerance 2). One key-value pair will specify the angle threshold for the head pose estimation to determine if mutual gaze is achieved. If the angle of the head pose is less than this threshold, it will be assumed that the person is looking at the robot (e.g., angleThreshold 15 indicates that angles below 15 degrees imply mutual gaze). One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true). One key-value pair will specify the algorithm to use (e.g., algorithm Haar | YOLO | CNN). The configuration file will be named faceDetectionConfiguration.json.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through ROS topics, services, or actions. In addition, it will include the specification of a driver to source test input data, and a specification of stub to sink test output data. Based on the module specification, it is planned that the interface of this module will use the physical robot as the primary driver to generate test data. Furthermore, it is planned to develop a driver to generate test data by acquiring RGB images from a USB camera or a USB RGB-D camera. This driver will publish the data on the same topics as those used by the physical robot. It is planned to develop an stub module to subscribe to the /faceDetection/data topic and write the 2D image location data to the screen.

The module design section will specify selected algorithms and data structures, and develop a unit test plan. It is planned to utilize the function for face and eye detection in OpenCV library which uses Haar features and boosted classification techniques.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

Description of Deliverables (continued)

The unit testing section will present the unit tests to verify that the module meets the module specification under different conditions, e.g., partial occlusion of the face and different lighting conditions. A unit test will be provided for each of the two test platforms: driver-stub unit test and physical robot unit test.

The unit tests will be invoked by a launch file that brings up the required resources, e.g., any software to communicate the physical robot or the driver and stub. The driver will publish the sensor data on the physical robot sensor topics. A test report will document the results of these tests.

In addition, a user manual will be included in the deliverable.

Note that this module does not support the simulator because it can optionally use an external RealSense camera which is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D4.2.3 Sound Detection and Localization
Deliverable type: software and report
This deliverable represents the outcome of Task 4.2.3. It comprises the documented software required to build and launch a module implemented as a single ROS node to detect and determine the direction of arrival of a conspicuous sound within the robot's hearing range. The ROS node will be named soundDetection.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in the task description, identifying gaps and misalignments with actual needs.

The module specification section will define the functional characteristics, as follows. The module will detect and determine the direction of arrival of a conspicuous sound within the robot's hearing range, and output the direction angle and the audio signal of the detected sound, from onset of the sound to offset of the sound. The module should be robust enough to operate in challenging acoustic conditions, e.g., in rooms with background noise or reverberation. Localization will be limited to the azimuth (i.e., horizontal) plane. If a sound is detected, its direction of arrival will be determined in the robot's Cartesian head frame of reference. The module will be tuned to detect human voices rather than ambient sounds or background noise by using signal processing techniques such as band-pass filtering and noise reduction. While the interaural time delay (ITD) between the arrival of the sound at the robot's microphones on the top of the robot's head is planned as the initial approach to be used to compute the direction of arrival of the sound, other techniques will also be investigated and evaluated, as required.

The input will take the form of a multichannel audio signal acquired using the robot's four microphones.

The output will be the angle of arrival of the sound in the robot's Cartesian head frame of reference, specified in degrees, and published on a topic named soundDetection/direction. In addition, the left channel of the audio signal that is captured will be published on a topic named soundDetection/signal. The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

The names of the topics to be used for each sensor will be read from a data file comprising a sequence of key-value pairs. The key is the name of the sensor. The value is the topic name.

The operation of the module will be determined by parameters provided in a configuration file that contains a list of key-value pairs. One key-value pair will specify the localization technique to be used (e.g., algorithm ITD).

Description of Deliverables (continued)

One key-value pair will specify the low cutoff frequency in the band-pass filter in hertz, e.g., lowFreqencyCutoff 100). One key-value pair will specify the high cutoff frequency in the band-pass filter in hertz, e.g., highFreqencyCutoff 15000). One key-value pair will specify the threshold energy of the audio signal that qualifies it as a conspicuous sound. e.g., thresholdEnergy 200). One key-value pair will specify the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true. The configuration file will be named soundDetectionConfiguration.ini.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through ROS topics, services, or actions. In addition, it will include the specification of a driver to source test input data, and a specification of stub to sink test output data. Based on the module specification, it is planned that the interface of this module will use the physical robot as the primary driver to generate test data. Furthermore, it is planned to develop a driver to generate test data either by acquiring audio signals from the stereo microphone input on the PC or from a data file. This driver will publish the data on the same topics as those used by the physical robot. It is planned to develop a stub module to subscribe to the /soundDetection/direction topic, and write the angle value to the screen.

The module design section will specify selected algorithms and data structures, and develop a unit test plan. It is planned to consider focus initially on interaural time difference (ITD) but other techniques will also be investigated and evaluated, as required.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will present the unit tests to verify that the module meets the module specification under different conditions, e.g., rooms with reverberation, background noise such as air conditioning, and the presence of background chatter. A unit test will be provided for each of the two test platforms: driver-stub unit test and physical robot unit test. The unit tests will be invoked by a launch file that brings up the required resources, e.g., any software to communicate the the physical robot or the driver and stub. The driver will publish the sensor data on the physical robot sensor topics. A test report will document the results of these tests.

In addition, a user manual will be included in the deliverable.

Note that this module does not support the simulator because the microphones are not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D4.2.4 Robot Localization
Deliverable type: software and report
This deliverable represents the outcome of Task 4.2.4. It comprises the documented software required to build and launch a module implemented as a single ROS node to determine the pose (position and orientation) of the robot in a Cartesian world frame of reference. The ROS node will be named robotLocalization.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in the task description, identifying gaps and misalignments with actual needs.

The module specification section will define the functional characteristics, as follows. This module will compute the pose of the robot in the world frame of reference. It will do this continuously, in real time, by updating the current pose based on relative pose estimation, using either odometry or the robot's inertial management unit IMU (or a combination of both). Since pose estimation errors using relative techniques grow with time, the module will periodically reset its pose estimate using absolute pose estimation. Position estimation will be accomplished by triangulation, using three landmarks (three are required because their distance from the robot is not known since the robot's range sensors are not sufficiently accurate). Landmark recognition will be accomplished either using SIFT (Scale Invariant Feature Transform) or YOLO (You Look Only Once) real-time object detection. The position of the landmarks will be extracted from a map of the environment. This map will be produced in Task 5.5.3 Environment Map Generation. The orientation of the robot will be computed only for its rotation about the Z-axis; adjustments of body posture through rotation about the X- and Y-axes will be ignored. This rotation angle will be recovered by determining the direction given by the line of sight from the robot to one of the landmarks, and adjusting for any rotation about the Z-axis of the robot's head frame of reference with respect to the base frame of reference.

For relative pose estimation, the input will be the odometry data published by the robot and data from the robot's accelerometer and gyrometer. For absolute pose estimation, the input will take the form of an RGB image from one of the robot's cameras and a depth image from one of the robot's depth sensors. Input will also be acquired from the encoder on the head yaw actuator, i.e., the joint responsible for rotation in the azimuth (horizontal) plane. The module can also service requests to reset the robot's pose using absolute pose estimation.

The output will take the form of an RGB image, with bounding boxes drawn around each detected landmark, a depth image, again with bounding boxes drawn around each detected landmark, and a record with the 2D pose information: x and y coordinates and rotation about the Z-axis. The record will be published on a topic named robotLocalization/pose_data. The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal. Also, if the module is operating in verbose mode, the RGB image will be displayed in an OpenCV window.

Description of Deliverables (continued)

The names of the topics to be used for each sensor will be read from a data file comprising a sequence of key-value pairs. The key is the name of the sensor. The value is the topic name.

The operation of the module will be determined by parameters provided in a configuration file that contains a list of key-value pairs. One key-value pair will specify the RGB camera to be used (e.g., camera FrontCamera | camera StereoCamera). One key-value pair will specify the distance that can be travelled in centimetres before the relative pose estimate is reset using the absolute pose estimate (e.g., resetInterval 300). One key-value pair will specify the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true). The configuration file will be named robotLocalizationConfiguration.ini.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through ROS topics, services, or actions. In addition, it will include the specification of a driver to source test input data, and a specification of stub to sink test output data. Based on the module specification, it is planned that the interface of this module will use the physical robot as the primary driver to generate test data. Furthermore, it is planned to develop a driver to generate test data by acquiring RGB images from a USB camera or a USB RGB-D camera, and acquiring depth images from a USB RGB-D camera. This driver will also publish synthetic odometry data and synthetic IMU data. It is planned to develop an OpenCV stub module to subscribe to the robotLocalization/pose_data topic and write the pose data to the screen.

The module design section will specify selected algorithms and data structures, and develop a unit test plan. It is planned to consider two options, i.e., SIFT and YOLO for landmark recognition.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will present the unit tests to verify that the module meets the module specification under different conditions, e.g., partial occlusion of the landmarks and different lighting conditions. This test will also include a comparison of the performance of the module with the localization performance of the SLAM (Simultaneous Localization and Mapping) module in Deliverable D5.5.3 Environment Map Generation. A unit test will be provided for each of the two test platforms: driver-stub unit test and physical robot unit test.

Description of Deliverables (continued)

The unit tests will be invoked by a launch file that brings up the required resources, e.g., any software to communicate the the physical robot or the driver and stub. The driver will publish the sensor data on the physical robot sensor topics. A test report will document the results of these tests.

In addition, a user manual will be included in the deliverable.

Note that this module does not support the simulator because it can optionally use an external RealSense camera which is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D4.2.5 Camera Calibration
Deliverable type: software and report
This deliverable represents the outcome of Task 4.2.5. It comprises the documented software required to build and launch a module implemented as a single ROS node to determine the 3×4 camera model matrix for the robot's front camera and the two stereo cameras. The ROS node will be named cameraModel.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in the task description, identifying gaps and misalignments with actual needs.

The module specification section will define the functional characteristics, i.e., the ability to determine the 3×4 camera model matrix for the robot's front camera and stereo cameras by extracting the 2D image control point coordinates from two or more images of a standard checkerboard calibration grid and computing the camera model based on the corresponding 3D world coordinates. The 3D coordinates of the calibration grid will be determined empirically, by placing it at a known position and orientation with respect to the robot head, and computing the coordinates from the spacing of the grid pattern and the grid pose. Thus, the 3D coordinates that are computed using the resultant inverse perspective transformation are defined with respect to the robot head frame of reference.

The input will take the form of RGB images from the robot's front and stereo cameras. In addition, the module will input from file the specification of the camera calibration setup (i.e., calibration grid pattern, spacing, position, and orientation), and the coordinates of the 3-D world control points, either all points or the four extreme corner points, with the remaining points being computed by bilinear interpolation.

The output will be a file containing the twelve floating point elements of the camera model matrix. The module can run in normal mode or verbose mode. In verbose mode, the camera model matrix is printed to the terminal. Also, if the module is operating in verbose mode, an RGB image with the calibration points graphically identified will be displayed in an OpenCV window.

The names of the topics to be used for each sensor will be read from a data file comprising a sequence of key-value pairs. The key is the name of the sensor. The value is the topic name.

The operation of the module will be determined by parameters provided in a configuration file that contains a list of key-value pairs. One key-value pair will specify the camera to be calibrated (e.g., camera FrontCamera | camera StereoCamera). One key-value pair will specify the number of views of the calibration grid to use in the calibration process (e.g., numberOfViews 2). One key-value pair will specify the filename of the .xml file that contains the specification of the camera calibration setup, including the size of the calibration grid, the grid spacing, and the calibration pattern (e.g., calibrationConfiguration cameraCalibConfig.xml).

Description of Deliverables (continued)

One key-value pair will specify the filename of the file that contains the coordinates of the corresponding 3-D world control points (e.g., worldControlPoints cameraModelWorldControlPoints.dat). One key-value pair will specify the filename of the file to which the twelve floating point elements of the camera model matrix are to be written. (e.g., cameraModel cameraModelCoefficients.dat). One key-value pair will specify the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true). The configuration file will be named cameraModelConfiguration.ini.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through ROS topics, services, or actions. In addition, it will include the specification of a driver to source test input data, and a specification of stub to sink test output data. Based on the module specification, it is planned that the interface of this module will use the physical robot as the primary driver to generate test data. Furthermore, it is planned to develop a driver to generate test data by acquiring RGB images from a USB camera and a USB RGB-D camera, in the case that the RGB camera is being calibrated, or just a USB RGB-D camera, in the case that the RGB-D camera is being calibrated.

The module design section will specify selected algorithms and data structures, and develop a unit test plan. It is planned to compute the camera model by computing a least-square-error solution to an over-determined system of equations derived by associating image control point coordinates, world control point coordinates, and the camera model coefficients.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will present the unit tests. One unit test will verify that the camera model can regenerate all the x and y coordinates of the world control points from the coordinates of the image control points and the z values of the world control point. One unit test will allow a user to interactively select an image point in the input RGB image and display the corresponding 3D world coordinates. This latter test will be provided for each of the two test platforms: driver-stub unit test and physical robot unit test. The unit tests will be invoked by a launch file that brings up the required resources, e.g., any software to communicate the the physical robot or the driver and stub. The driver will publish the sensor data on the physical robot sensor topics. A test report will document the results of these tests.

In addition, a user manual will be included in the deliverable.

Note that this module does not support the simulator because it can optionally use an external RealSense camera which is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D4.3.1 Tablet PC Event
Deliverable type: software and report
This deliverable represents the outcome of Task 4.3.1. It comprises the documented software required to build and launch a module implemented as a ROS node that provides a ROS service to display a menu of interaction options that are appropriate in the current context of the scenario, wait for the visitor to select an option, and return the option selected. This service will be called by the `behaviorController`, in the interaction manager subsystem. The ROS node will be named tabletEvent.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in this work plan, identifying gaps and misalignments with actual needs.

The module specification section will specify the functional characteristics, detailing the input to output data transformation, expected input, expected output data, module configuration parameters, and a draft user manual. Functionally, this module will display a menu of options that the user can select. The menu is specified in the ROS service request. The module will block until the user touches the table PC screen, and, depending on the area touched, the module will unblock and respond to the service with the identifier of the selected option. The module can run in normal mode or verbose mode. In verbose mode, menu and response are also printed to the terminal. The operation of the module will be determined by parameters provided in a configuration file. These parameters comprise a list of key-value pairs. One key-value pair will specify the filename of the menu data (e.g., menuDataFile tabletEventInput.dat). One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true). The module's application programming interface (API), i.e., the invocation and control of the module, will be documented in a user manual.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed through ROS services and files. In addition, it will include the specification of a driver to request this service, and display the menu option selected by the visitor with whom the robot is interacting.

The module design section will specify the selected algorithm and data structure used to match the screen region to the menu option identifier, and develop a unit test plan.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

Description of Deliverables (continued)

The unit testing section will provide a verification unit test to ensure that the correct menu option identifier is used in the service response. A test report will document the results of these tests.

In addition, a user manual will be included in the deliverable.

Note that this module does not support the simulator because the tablet PC is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D4.3.2 Speech Event
Deliverable type: software and report
This deliverable represents the outcome of Task 4.3.2. It comprises the documented software required to build and launch a module implemented as a single ROS node that detects an utterance, represented as an audio signal, spoken by an interaction partner in Kinyarwanda or English, and transcribes it into written text. The ROS node will be named speechEvent.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in this work plan, identifying gaps and misalignments with actual needs.

The module specification section will specify the functional characteristics, detailing the input to output data transformation, expected input data, expected output data, module configuration parameters, and a draft user manual. Functionally, this module first requires that a deep neural network be trained and tested so that it can perform automated speech recognition. This network will subsequently be used to take as input the audio signal of a sound detected by the module developed in Task 4.2.3 Sound Detection and Localization, and produce the corresponding written text. In the case that the spoken utterance cannot be recognized, either because the sound is not a spoken utterance or because it uses vocabulary on when the neural network has not been trained, then the module will flag this by producing a text that reads "Error: speech not recognized".

The module can run in normal mode or verbose mode. In verbose mode, menu and response are also printed to the terminal. The operation of the module will be determined by parameters provided in a configuration file. These parameters comprise a list of key-value pairs. One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true).

The input will take the form of an audio signal that is captured in the soundDetection module, published on a topic named soundDetection/signal.

The output will take the form of a record containing a string representing the message in the spoken audio signal. This will be published on a topic named speechEvent/text. The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

Description of Deliverables (continued)

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through ROS topics, services, or actions. In addition, it will include the specification of a driver to source test input data. Based on the module specification, it is planned that the interface of this module will develop a driver to generate test data based on parameters defined in the driver input file.

The module design section will comprise a complete and comprehensive description of the deep neural network and the training procedure.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will present the unit tests. At least two types of unit test will be delivered: (i) verification against module specification, and (ii) validation against requirements definition. Where feasible, an evaluation unit test against benchmark data or standard metrics will be delivered. A test report will document the results of these tests.

In addition, a user manual will be included in the deliverable.

Note that this module does not support the simulator because it is dependent on input from sound detection and localization which does not support the simulator; see the notes on Version 3.0 in the Document History.

D4.4 Use Case Feedback
Deliverable type: report.
This deliverable represents the outcome of Task 4.4. It documents any adjustments that need to be made to the robot sensing based on the evaluation of the two use cases in work package WP6.

2.4.5 Work Package 5: Robot Behaviors

Objectives

1. Develop a suite of unit and system tests to demonstrate that all actuators are functioning correctly and can be controlled using ROS.
2. Develop an animate behavior subsystem comprising software modules that enable the robot to exhibit life-like characteristics which encourage interaction by the human partner.
3. Develop an attention sub-system comprising software modules that controls the robot's overt attention.
4. Develop an interaction manager subsystem comprising a software module that controls robot interaction with a human partner, taking into consideration culturally sensitive behaviors for Africa and the requirements of a use case scenario.
5. Develop a subsystem comprising software modules for robot gestures, speech, and navigation.

Description of Work

Task 5.1: Actuator Tests (M1–M4)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to write a suite of unit tests, one for each actuator, to verify that every actuator can be controlled successfully using ROS. A subset of the unit tests can be selected, and those that are selected can be run in sequence or in parallel.

This objective will be achieved by creating a list of actuators and the associated topics used by the physical Pepper robot and the Pepper robot simulator, writing a suite of unit tests that will verify the operation of each sensor on both the physical Pepper and the simulator, writing a unit test scheduling program that will use an external configuration file to identify the tests to be run, the manner in which they are to be run (in sequence or in parallel), and the platform on which they are to be run (the physical robot or the simulator). In the case of joint actuators, the unit test should move each joint to its minimum, maximum, and mid-range position (in that order) at a selected average angular velocity. In the case of wheel actuators, there will be two unit tests. The first unit test should rotate the robot at a selected positive angular velocity, e.g., $90°/s$, for a fixed period of time, e.g., one second, and it should then rotate the robot at a selected negative angular velocity, e.g., $-90°/s$, for a fixed period of time. The second unit test should move the robot at a selected positive linear velocity, e.g., 1m/s, for a fixed period of time, e.g., one second, and it should then move the robot at a selected negative linear velocity, e.g., -1m/s, for a fixed period of time. These unit tests should work with both the physical robot and the robot simulator.

The software development process for each unit test and the scheduling program will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are detailed in Deliverable D5.1.

Description of Work (continued)

Task 5.2: Animate Behavior Subsystem (M5–M9)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to develop a software module to give the robot the appearance of an animate agent by continually making subtle body movements, flexing its hands a little, and rotating its base slightly.

This objective will be achieved by actuating the robot joints periodically in some random pattern, keeping the joint angles close to the default home positions. The extent of the movements should be determined by an external parameter. Since the robot should not make these animate movements when engaged in culturally sensitive social interaction through gestures, speech, or when navigating, the generation of animate behaviour will be started and stopped by the Interaction Manager subsystem. Note that, from a software architecture perspective, the Attention subsystem may also contribute to the appearance of an animate agent by directing the robot gaze to relevant objects, even when not engaging in social interaction. The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are detailed in Deliverable description D5.2 Animate Behavior Subsystem. The task uses the outcome of Task 5.1 Actuator Tests as documented in Deliverable D5.1.

Task 5.3: Attention Subsystem (M7–M15)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to develop a software module to allow 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 bodies, faces, eyes, and voices. These features will also be 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 (Task 5.2). Observing the environment also involves scanning the environment for interesting objects. For this, a more general-purpose saliency function is required to complement the social interaction feature saliency function. Furthermore, scanning the environment requires the robot to change its focus of attention after some time and not return directly to the original focus of attention.

This objective will be achieved by generating two saliency maps, one based on social features and one based on general interesting features. The former will use the output of Task 4.2.2 Face & Mutual Gaze Detection and Localization and Task 4.2.3 Sound Detection and Localization. Candidates for the latter include the Itti and Koch saliency model of visual attention (Itti et al., 1998; Itti & Koch, 2000, 2001), the model proposed by Rea et al. (2013), the information-theoretic saliency map (Bruce & Tsotsos, 2009), and open source deep neural network saliency models. It will also make use of the output of Task 4.2.2 Face & Mutual Gaze Detection and Localization, such that faces can be prioritized as salient points in the environment on top of the general saliency function. A single focus of attention will be selected from the saliency map with a winner-take-all process. A habituation process reduces the salience of the current focus of attention with time, thereby ensuring that attention is fixated on a given point for a limited period. An Inhibition-of-Return (IOR) mechanism will be developed to attenuate the attention value of previous winning locations so that new regions become the focus of attention. Furthermore, attention will also be directed toward conspicuous sounds. Aural attention will have a higher priority than visual attention. For direction of attention at a given location in the environment, the pose of the robot in the world frame of reference will also be used.

Description of Work (continued)

The robot's gaze will be adjusted so that the gaze is centred on the focus of attention. This requires the calibration of the x and y offset of the focus of attention in the image to the change in headYaw and headPitch angles, respectively. For simplicity, it will be assumed that there is a linear relationship between these two variables. In the case of aural attention to conspicuous sounds, it requires the calibration of the angle of arrival of the sound with the change in headYaw angle. Fixation on sounds will only control the headYaw angle, i.e., rotation in the horizontal plane about the head's Z-axis. If the angle of rotation of the headYaw is greater than some given threshold, then after rotating the head to fixate on the focus of attention, the base of the robot and the head will rotate in opposite directions so that the robot continues to gaze at the focus of attention as it realigns its head with its body.

The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are detailed in Deliverable D5.2 Attention. The task uses the output of Task 4.2.2 Face & Mutual Gaze Detection and Localization and Task 4.2.3 Sound Detection and Localization.

Note that this module does not support the simulator because it can optionally use an external RealSense camera which is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Task 5.4: Interaction Manager Subsystem (M7–M18)
Responsible Partner: University of the Witwatersrand.
The Interaction Manager Subsystem will orchestrate the way the Pepper robot interacts with visitors in the two demonstration scenarios, using culturally sensitive spatial, nonverbal, and speech actions. There are four elements to this task:

1. The development of a cultural knowledge ontology.
2. The population of a knowledge base that adheres to the structure given by the ontology.
3. The definition of a scenario robot mission language that can capture the interactions in different social robotics scenarios, and the two selected use case scenarios, in particular.
4. The specification, design, and implementation of an interpreter for the robot mission language that can transform the interactions specified in the use case scenarios into a sequence of culturally sensitive human-robot interactions.

Elements 1 and 2 are addressed in Task 5.4.1, effectively formalizing the knowledge that will have been acquired in work package WP1. Element 3 is addressed in Task 5.4.2 and element 4 is addressed in Task 5.4.3.

Description of Work (continued)

Task 5.4.1: Cultural Knowledge Ontology & Culture Knowledge Base (M10–M18)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to formulate a cultural knowledge ontology, use it to create a knowledge base formalizing the culturally sensitive knowledge that are documented in Deliverables D1.2 African Modes of Social Interaction and D1.3 Africa-centric Design Patterns, and create a knowledge management software module to act as a server for this knowledge base.

This objective will be achieved by mapping the knowledge in Deliverable D1.2 to the cultural knowledge ontology proposed by Bruno et al. (2019), adapting the ontology and extending it, as necessary, based both on the other ontologies in the literature and the requirements of cultural knowledge in Africa, specifically in South Africa and Rwanda. This will facilitate the development of a structured knowledge base that encapsulates the knowledge captured in Deliverable D1.3. The outcome of this task is described in Deliverable D5.4.1.

Task 5.4.2: Robot Mission Language (M7–M12)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to define a language that can be used to capture the interactions in the use case scenarios and enact them using the cultural knowledge ontology and culture knowledge base.

This objective will be achieved by modelling the use case scenarios with finite state machines (FSM), also known as state transition diagrams, and mapping these to an associated finite state language and grammar. The task uses the outcome of Task 2.1, documented in Deliverable D2.1, by recruiting the robot and visitor behaviours documented in Deliverables D2.2 and D2.3. It also uses the cultural knowledge ontology and culture knowledge base developed in Task 5.4.1, encapsulated in Deliverable D5.4.1. The outcome of this task is described in Deliverable D5.4.2.

Task 5.4.3: Robot Mission Interpreter (M13–M18)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to develop an interpreter for the robot mission language defined in Deliverable D5.4.2.

This objective will be achieved either by developing a domain specific language (DSL), based on the language grammar, to implement a finite state machine (FSM) model of the interaction, or by adapting an open source implementation of a FSM language interpreter.

Note that this module does not support the simulator because it is dependent on input from person detection, robot localization, and speech event, all of whihc do not support the simulator; see the notes on Version 3.0 in the Document History.

Description of Work (continued)

Task 5.5: Expression & Actuation Subsystem (M1–M18)
Responsible Partner: Carnegie Mellon University Africa
The Expression and Actuation Subsystem enables the robot to interact with people in a culturally sensitive manner, and navigate its environment while doing so. It also implements the animate behavior actuation when the robot is idle, i.e., when it is not engaged in interaction. It comprises three interaction tasks, one concerned with non-verbal gestural communication, one with speaking to people, and one with navigating the environment, adhering to cultural norms as it moves. The latter has an associated task concerned with creating a map of the environment that the robot has to navigate.

Task 5.5.1: Gesture Execution (M4–M15)
Responsible Partner: Carnegie Mellon University Africa.
The objective of the Gesture Execution task is to develop a software module that 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.

This objective will be achieved by identifying a way of specifying all five forms of gesture. The specifications for these gestures will be in joint space, except for deictic gestures which will be in Cartesian space. Some gestures, e.g., iconic and symbolic hand gestures will be specified by learning the required motions either by recording the joint angles during manual teleoperation, or by demonstration, using an RGB-D depth camera to determine the joint angles of human gestures in a skeletal model and mapping these to the robot joints. Other gestures, i.e., deictic hand gestures and body gestures, will be specified by gesture parameters, such as the pointing location for deictic gestures and the degree of inclination for bowing and nodding, and the joint angles will be computed using the kinematic model of the robot head, torso, and arms. For deictic gestures, which require the robot to point and look at objects in its environment, the pose of the robot in the world frame of reference will also be used.

The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are detailed in Deliverable D5.5.1.1 Gesture Execution and D5.5.1.2 Programming by Demonstration.

The task uses the outcome of Task 4.2.4 Robot Localization as documented in Deliverable D4.2.4.

Note that this module does not support the simulator because it is dependent on input from robot localization which does not support the simulator; see the notes on Version 3.0 in the Document History.

Description of Work (continued)

Task 5.5.2: Text to Speech Conversion (M7–M12)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to implement a text to speech facility that will allow the Pepper robot to communicate with humans using spoken language. The primary language of communication will be English. In anticipation of more natural interaction in the future, we will also address text to speech for isiZulu and Kinyarwanda speakers, so that Pepper will be able to use its voice to provide spoken greetings and responses in isiZulu and Kinyarwanda. The possibility of controlling loudness, pitch, and timbre of the speech will be investigated.

This objective will be achieved by investigating the feasibility of using the NAOqi audio utility ALTextToSpeech through a ROS interface. In parallel, we will investigate the development of a speech synthesis engine for isiZulu and Kinyarwanda, possibly leveraging work by ? (?).

To facilitate the Afretec requirement that only one partner university can be assigned to each activity, i.e., task, Task 5.5.2 comprises four sub-tasks, one for English text-to-speech, one for isiZulu text-to-speech, and one for Kinyarwanda text-to-speech, and one to integrate all three, each with corresponding deliverables. Note that the integrated text-to-speech functionality will encapsulate the other three in a single ROS node.

These tasks use the outcome of Task 5.4.3 as documented in Deliverable D5.4.3 Robot Mission Interpreter.

Task 5.5.2.1: English Text to Speech Conversion (M7–M12)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to implement an English text to speech facility that will allow the Pepper robot to communicate with humans using spoken English.

The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are described in Deliverable D5.5.2.1.

Note that this module does not support the simulator because the speakers are not available in Gazebo using the `\speech` topic; see the notes on Version 3.0 in the Document History.

Task 5.5.2.2: isiZulu Text to Speech Conversion (M7–M12)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to implement an isiZulu text to speech facility that will allow the Pepper robot to communicate with humans using spoken isiZulu.

The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are described in Deliverable D5.5.2.2.

Note that this module does not support the simulator because the speakers are not available in Gazebo using the `\speech` topic; see the notes on Version 3.0 in the Document History.

Description of Work (continued)

Task 5.5.2.3: Kinyarwanda Text to Speech Conversion (M7–M12)
Responsible Partner: Carnegie Mellon University.
The objective of this task is to implement an Kinyarwanda text to speech facility that will allow the Pepper robot to communicate with humans using spoken Kinyarwanda.

The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are described in Deliverable D5.5.2.3.

Note that this module does not support the simulator because the speakers are not available in Gazebo using the `\speech` topic; see the notes on Version 3.0 in the Document History.

Task 5.5.2.4: Integrated Text to Speech Conversion (M13–M15)
Responsible Partner: Carnegie Mellon University.
The objective of this task is to implement an integrated text to speech facility that will allow the Pepper robot to communicate with humans using spoken Englihs, isiZulu, and Kinyarwanda.

The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are described in Deliverable D5.5.2.4.

This task uses the outcome of Tasks 5.5.2.1, 5.5.2.2, and 5.5.2.3, as documented in Deliverables D5.5.2.1, D5.5.2.2, and D5.5.2.3.

Note that this module does not support the simulator because the speakers are not available in Gazebo using the `\speech` topic; see the notes on Version 3.0 in the Document History.

Task 5.5.3: Environment Map Generation (M1–M9)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to develop a software module to generate a metric map of the environments that will be used in the two use case scenarios documented in Deliverable D2.1. This map will be used by the Pepper robot when navigating and when making deictic gestures, i.e., when pointing at items of interest to the human in the use case scenario. As such, the map will not only compute the environment workspace space (i.e., the locations that are not occupied by obstacles) but also the locations and identities of objects of interest, as specified in the use case scenarios. Thus the map will comprise both non-symbolic metric data and symbolic semantic data.

This objective will be achieved by pursuing two approaches: (a) map generation from a priori data (i.e., environment CAD data), and (b) map generation using Simultaneous Localization and Mapping (SLAM). Both will generate non-symbolic metric map data that can be visualized as an image, and a list of the locations of labelled objects of interest, interactively selected by a human map builder.

Description of Work (continued)

The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are described in Deliverable D5.5.3.

The task uses the outcome of Task 5.4.3 as documented in Deliverable D5.4.3 Robot Mission Interpreter.

Task 5.5.4: Robot Navigation (M10–M18)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to develop a software to control the locomotion of the Pepper robot so that it can navigate in its environment, in which there are fixed inanimate obstacles and moveable obstacles in the form of humans, from its current position to a target destination along the shortest path. Navigation will be effected by identifying waypoints along the navigation path and the robot will move from waypoint to waypoint.

This objective will be achieved by first augmenting a metric workspace map of the robot's environment, i.e., the outcome of Task 5.5.3, with obstacles corresponding to the location of any humans that have been detected in the robot's field of view. The size of the human obstacle will be determined using culturally sensitive proxemics. This augmented workspace map will then be used to generate a configuration space map (i.e., the positions that robot can actually occupy, taking into consideration the size and shape of its base) that constrains the robot's path from its current location to its target location. The shortest path will then be computed using one of two shortest path algorithms: Dijkstra's algorithm and the A* algorithm. Waypoints will be identified using one of two candidate techniques: equidistant waypoints and high path curvature waypoints. Locomotion from waypoint to waypoint will be effected using one of two locomotion algorithms, such as Multiple Input Multiple Output (MIMO) and divide and conquer (DnQ).

The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are described in Deliverable D5.5.3.

The task uses the information produced by the modules in Deliverables D1.2 African Modes of Social Interaction, D2.1 Use Case Scenario Definition, D4.2.1 Person Detection and Localization, and D5.5.3 Environment Map Generation.

Note that this module does not support the simulator because it is dependent on input from person detection and localization which does not support the simulator; see the notes on Version 3.0 in the Document History.

Task 5.6: Use Case Feedback (M25–M33)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to update the robot behaviors based on feedback from the use case evaluations. This objective will be achieved by identifying any adjustments that are necessary to improve the acceptability, performance, and cultural sensitivity of the use cases. The task uses the outcomes of Task 6.2, as documented in Deliverable D6.2, The outcome of this task is summarized in Deliverable D5.6 and incorporated in revised versions of Deliverables D5.2, D5.3, D5.4.1 – D5.4.3, and D5.5.1 – D5.5.4.

List of Deliverables

Number	Deliverable Title	Responsible Partner	Delivery Month
D5.1	Actuator Tests	CMU-Africa	M4
D5.2	Animate Behavior Subsystem	CMU-Africa	M9
D5.3	Attention Subsystem	CMU-Africa	M15
D5.4.1	Cultural Knowledge Ontology & Culture Knowledge Base	Wits	M18
D5.4.2	Robot Mission Language	Wits	M12
D5.4.3	Robot Mission Interpreter	Wits	M18
D5.5.1.1	Gesture Execution	CMU-Africa	M15
D5.5.1.2	Programming by Demonstration	CMU-Africa	M18
D5.5.2.1	English Text to Speech Conversion	Wits	M12
D5.5.2.2	isiZulu Text to Speech Conversion	Wits	M12
D5.5.2.3	Kinyarwanda Text to Speech Conversion	CMU-Africa	M12
D5.5.2.4	Integrated Text to Speech Conversion	Wits	M15
D5.5.3	Environment Map Generation	CMU-Africa	M9
D5.5.4	Robot Navigation	CMU-Africa	M18
D5.6	Use Case Feedback	CMU-Africa	M33

Description of Deliverables

D5.1 Actuator Tests
Deliverable type: software and report
This deliverable represents the outcome of Task 5.1. It comprises the documented software required to build and launch a module implemented as a single ROS node that encapsulates a suite of unit tests to verify that each actuator can be successfully controlled. The ROS node will be named actuatorTest.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in the task description, identifying gaps and misalignments with actual needs.

The module specification section will define the functional characteristics, as follows.

In the case of joint actuators, the unit test will move each joint to its minimum, maximum, and mid-range position (in that order) at a selected average angular velocity. The actuator can then be controlled by specifying the required duration Δt for the total rotation, viz. $\Delta t = {}^{\Delta\theta}/_{\dot{\theta}}$, where $\Delta\theta$ is the total change in actuator angle, and $\dot{\theta}$ is the selected average angular velocity. In the case of wheel actuators, there will be two unit tests. The first unit test will rotate the robot at a selected positive angular velocity, e.g., $90°/s$, for a fixed period of time, e.g., one second, and it will then rotate the robot at a selected negative angular velocity, e.g., $-90°/s$, for a fixed period of time. The second unit test will move the robot at a selected positive linear velocity, e.g., 1m/s, for a fixed period of time, e.g., one second, and it will then move the robot at a selected negative linear velocity, e.g., -1m/s, for a fixed period of time. These unit tests should work with both the physical robot and the robot simulator.

Description of Deliverables (continued)

The names of the topics to be used for each actuator will be read from a data file comprising a sequence of key-value pairs. The key is the name of the actuator. The value is the topic name. There will be two data files, one for the physical robot and another for the simulator.

The actuators to be tested are to be identified by reading an associated input file that contains a list key-value pairs. The keys will be the the actuator groups: Head (HeadPitch and HeadYaw actuators), RArm (ShoulderRoll, ShoulderPitch, ElbowRoll, ElbowYaw, WristYaw, and Hand actuators), LArm (ShoulderRoll, ShoulderPitch, ElbowRoll, ElbowYaw, WristYaw, and Hand actuators), Leg (HipRoll, HipPitch, and KneePitch actuators), and Wheels (WheelB, WheelFR, and WheelFL actuators). The values will be either True or False, e.g., RArm True. The actuator group will be tested if the value is True. The keys and key values are not case sensitive. Note that the average velocities are not specified in the input file but are hard-coded. The input data file will be named actuatorTestInput.dat.

The operation of the module will be determined by parameters provided in a configuration file that contains a list of key-value pairs. One key-value pair will specify the platform on which the tests are to be run, i.e., the physical Pepper robot or the Pepper simulator (e.g., platform robot | platform simulator). One key-value pair will specify the mode in which the unit tests should be run: in sequence or in parallel (e.g., mode sequential | mode parallel). One key-value pair will specify the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). One key-value pair will specify the filename of the file in which the simulator sensor and actuator topic names are stored (e.g., simulatorTopics simulatorTopics.dat). The configuration file will be named actuatorTestConfiguration.ini.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through files, or ROS subscribers, services, or actions. Based on the module specification in the task description, it is planned that the interface of this module will use both the physical robot and the simulator to sink the control data published on the various actuator topics.

The module design section will specify selected message data structures used to represent and store the topic messages required to control the actuators.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will document the results of these using this actuatorTest module.

Finally, a user manual will be included in the deliverable, explaining how to invoke, provide input to, and configure the actuatorTest module.

Description of Deliverables (continued)

D5.2 Animate Behavior Subsystem
Deliverable type: software and report
This deliverable represents the outcome of Task 5.2. It comprises the documented software required to build and launch a module implemented as a single ROS node that 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. The ROS node will be named animateBehavior.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in the task description, identifying gaps and misalignments with actual needs.

The module specification section will define the functional characteristics, i.e., actuating the robot joints periodically in some random pattern, keeping the joint angles close to the default home positions. All the joints, except headYaw and headPitch (which are controlled by the Attention subsystem), will be actuated, as well as the wheels to effect rotation about the robot's Z-axis but not forward movement along the robot's X-axis. The extent of the movements will be determined by an external parameter. Specifically, the range of movement, from which the actual movement will be a random sample, will be specified as a percentage of half the full range of movement. Thus, 50% would mean that the amount of motion would vary randomly between the home value and half the maximum value, assuming the home value is midway between the minimum and maximum values.

The names of the topics to be used for each actuator will be read from a data file comprising a sequence of key-value pairs. The key is the name of the actuator. The value is the topic name. There will be two data files, one for the physical robot and another for the simulator.

Any of the three types of animate behavior — body movement, hand flex, and rotation — can be selectively invoked. All three will be invoked if none are selectively invoked.

To ensure that the robot does not make these animate movements when engaged in culturally sensitive social interaction through gestures, speech, or when navigating, the generation of animate behaviour will be started and stopped by the Interaction Manager subsystem using a dedicated service advertized and served by this node.

The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

Description of Deliverables (continued)

The operation of the animateBehavior module is determined by the contents of a configuration file that contain a list of key-value pairs. One key-value pair will specify the platform on which the tests are to be run, i.e., the physical Pepper robot or the Pepper simulator (e.g., platform robot | platform simulator). Up to three key-value pairs can be provided to identify the type of animate behavior to exhibit (e.g., behavior body | behavior hands | behavior rotation). One key-value pair will specify the range of actuator movement, specified as a percentage of half the full range of movement (e.g., range 5). One key-value pair will specify the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). One key-value pair will specify the filename of the file in which the simulator sensor and actuator topic names are stored (e.g., simulatorTopics simulatorTopics.dat). One key-value pair will specify whether diagnostic data is to be printed to the terminal (e.g., verboseMode true). The configuration file will be named animateBehaviorConfiguration.ini.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through files, or ROS subscribers, services, or actions. Based on the module specification in the task description, it is planned that the interface of this module will use both the physical robot and the simulator to sink the control data published on the various actuator topics.

The module design section will specify selected message data structures used to represent and store the topic messages required to control the actuators.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will present the unit tests to verify that the module meets the module specifications. A unit test will be provided for each of the two test platforms: (i) physical robot unit test, and (ii) simulator unit test. The unit tests will be invoked by a launch file that brings up the required resources, e.g., any software to communicate the the physical robot, the simulator. For the physical robot or simulator unit tests to work correctly, the node configuration file (*.ini) has to have the correct platform selected. A test report will document the results of these tests.

Finally, a user manual will be included in the deliverable, explaining how to invoke, provide input to, and configure the animateBehavior module.

Description of Deliverables (continued)

D5.3 Attention Subsystem
Deliverable type: software and report
This deliverable represents the outcome of Task 5.3. It comprises the documented software required to build and launch a module implemented as a single ROS node that provides the robot with the ability to pay attention to salient features in the environment or to a given location in the environment. The ROS node will be named overtAttention.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in the task description, identifying gaps and misalignments with actual needs.

The module specification section will define the functional characteristics, i.e., direct the gaze of the robot at salient features associated with social interaction, e.g., people's bodies, faces, eyes, and voices, both when the robot is interacting with people and when it is just observing its environment. The gaze can also be directed at a given location in the environment. If the head cannot achieve the required pose to gaze at a given location, the robot should rotate to make the pose achievable. When observing the environment, the robot will also fixate on other salient features or objects. In this mode of operation, the robot will change its focus of attention after some time and not return directly to the original focus of attention. Two saliency maps will be generated, one based on social features only, and one based on general interesting features and social features. The former will use the output of Task 4.2.2 Face & Mutual Gaze Detection and Localization and Task 4.2.3 Sound Detection and Localization. The latter uses the output of Task 4.2.2 Face & Mutual Gaze Detection and Localization for social features and candidates for the general interesting features include the model proposed by Rea et al. (2013), the information-theoretic saliency map (Bruce & Tsotsos, 2009), and open source deep neural network saliency models.

The complete overt attention system will comprise (a) a winner-take-all process effected by a selective tuning model to identify a single focus of attention (Tsotsos et al., 1995; Tsotsos, 2006, 2011), (b) an Inhibition-of-Return (IOR) mechanism to attenuate the attention value of previous winning locations so that new regions become the focus of attention, and (c) a habituation process to reduce the salience of the current focus of attention with time thereby ensuring that attention is fixated on a given point for a limited period (Zaharescu et al., 2004). In addition to gazing at people's faces, the attention system will also detect when a person close by is gazing directly at the robot. This allows for a simple form of mutual gaze detection. Furthermore, attention will be directed toward conspicuous sounds. Aural attention will have a higher priority than visual attention.

Description of Deliverables (continued)

Overall, the overt attention system will be in one of five modes at any given time, namely scanning mode, social mode, seeking mode, location mode, or disabled mode. In scanning mode the robot will gaze at generally salient features and people's faces in the environment. Also, in scanning mode the node will keep track of when people gaze directly at the robot and publish this through a topic to facilitate the beginning of an interaction (or tour) with seemingly interested people. In social mode the gaze will be directed to people's faces and voices to facilitate social interaction. In seeking mode, the robot will direct its attention to human faces, and it will detect when the human face is gazing directly at the robot. This mode is time-bound so that it will fail if mutual gaze is not detected in a given interval of time. In location mode the robot will fixate its gaze at a point in the environment. In disabled mode the attention system will center the head and then be put to sleep for few seconds repeatedly.

The robot's gaze will be directed by publishing the appropriate control messages on the headYaw and headPitch actuator topics so that the gaze is centred on the focus of attention. This requires the calibration of the x and y offset of the focus of attention in the image to the change in headYaw and headPitch angles, respectively. In the case of aural attention to conspicuous sounds, it requires the calibration of the angle of arrival of the sound with the change in headYaw angle. Fixation on sounds will only control the headYaw angle, i.e., rotation in the horizontal plane about the head's Z-axis. These calibration constants will be provided as parameters to the module. If the angle of rotation of the headYaw is greater than some threshold defined as a parameter, then after rotating the head to fixate on the focus of attention, the base of the robot and the head will rotate in opposite directions so that the robot continues to gaze at the focus of attention as it realigns its head with its body. The threshold for this realignment will be provided as a parameter to the module.

One input will take the form of an RGB image from one of the robot's cameras. Other inputs are the outputs of the modules in deliverables D4.2.2 Face & Mutual Gaze Detection and Localization and D4.2.3 Sound Detection and Localization. When attending to a given location in the environment, the module will also input the current robot pose from the robotLocalization module.

One output will take the form of an RGB image depicting the saliency function and the selected focus of attention. A second output is the current mode of the attention node — scanning, social, seeking, location, and disabled — published periodically on a dedicated topic. Other outputs are the control messages, published on the appropriate topics, to adjust the headYaw, headPitch, and, if required, the orientation of the robot.

The names of the topics to be used for each actuator will be read from a data file comprising a sequence of key-value pairs. The key is the name of the actuator. The value is the topic name.

The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal. Also, if the module is operating in verbose mode, the saliency image will be displayed in an OpenCV window.

Description of Deliverables (continued)

The operation of the attention module is determined by the contents of a configuration file that contains a list of key-value pairs. One key-value pair will specify the RGB camera to be use (e.g., camera FrontCamera | camera StereoCamera). One key-value pair will specify the threshold on the angular difference between head and base that must be met before the head and base are realigned (e.g., realignmentThreshold 5). One key-value pair will specify the calibration constant that defines the conversion of the offset in the X- (horizontal) axis of an image from the image center to the change in headYaw angle (e.g., xOffsetToHeadYaw 25). One key-value pair will specify the calibration constant that defines the conversion of the offset in the Y- (vertical) axis of an image from the image center to the change in headPitch angle (e.g., yOffsetToHeadPitch 20). One key-value pair will specify the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true). The configuration file will be named overtAttentionConfiguration.ini.

The names of the topics to be used for each actuator will be read from a data file comprising a sequence of key-value pairs. The key is the name of the actuator. The value is the topic name. The input data file for the physical robot will be named pepperTopics.dat.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through files, or ROS subscribers, services, or actions.

Based on the module specification in the task description, it is planned to develop a driver to generate test input data, invoking the faceDetection and soundDetection modules, and, in the event that the Pepper robot is not available for testing, acquiring the RGB image from an external camera.

The module design section will specify selected message data structures used to represent and store the topic messages required to control the actuators, and the image data structures, and saliency, focus of attention selection, habituation, and inhibition-of-return algorithms.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

Description of Deliverables (continued)

The unit testing section will present the unit tests to verify that the module meets the module specification under different conditions, e.g., different lighting conditions. A unit test will be provided for each of the two test platforms: driver-stub unit test and physical robot unit test. The unit tests will be invoked by a launch file that brings up the required resources, e.g., any software to communicate the the physical robot or the driver and stub. The driver will publish the sensor data on the physical robot sensor topics. A test report will document the results of these tests.

Finally, a user manual will be included in the deliverable, explaining how to invoke, provide input to, and configure the overtAttention module.

Note that this module does not support the simulator because it can optionally use an external RealSense camera which is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D5.4.1 Culture Knowledge Ontology & Culture Knowledge Base
Deliverable type: software and report
This deliverable represents the outcome of Task 5.4.1. It comprises three elements: (i) a cultural knowledge ontology, (ii) a related culture knowledge base file, and (iii) the documented software required to compile a C++ helper class to read the culture knowledge base file, store the knowledge, and make the knowledge accessible through a suite of access methods. In effect, this knowledge base formalizes the culturally sensitive knowledge that is documented in Deliverables D1.2 African Modes of Social Interaction. It provides the basis for generating culturally sensitive actions by the robot when interacting with visitors in the use case demonstrations. The C++ helper class will be named CultureKnowledgeBase.

In addition to functional code, the deliverable will include a report with sections detailing the specification, design, and implementation of the C++ helper class.

The population of the knowledge base, i.e., a C++ object instantiation of the CultureKnowledgeBase class, is determined by the contents of a configuration file that contain a list of key-value pairs. One key-value pair will specify the filename of the file in which the knowledge is stored. One key-value pair will specify whether diagnostic data is to be printed to the terminal (e.g., verboseMode true). Others will be added, as required. The configuration file will be named cultureKnowledgeBaseConfiguration.ini.

The module design section will specify selected class data structures used to represent and store the knowledge read from the culture knowledge base file.

Code will be written and documented in adherence to the standards set out in Deliverable D3.2 Software Engineering Standards Manual.

A unit test will be provided to verify that the class meets the requirements and adheres to the specifications. It will do this by instantiating a C++ helper class object, and using the access methods to retrieve and write the contents of the culture knowledge base file to the terminal.

Description of Deliverables (continued)

D5.4.2 Robot Mission Language
Deliverable type: software and report
This deliverable represents the outcome of Task 5.4.2. It comprises four elements: (i) a language or other mode of abstract modelling that can be used to formally specify the interactions in the use case scenarios and enact them in a culturally sensitive manner using the culture knowledge base and an environment knowledge base, (ii) two files containing a specification of the two use case scenarios in that language or abstract model, (iii) an environment knowledge base file with the information required to complete the robot mission, and (iv) the the documented software required to compile a C++ helper class to read the environment knowledge base file, store the knowledge, and make the knowledge accessible through a suite of access methods. As such, it provides the input for the development in Task 5.4.3 of an interpreter that can translate this abstract specification into robot actions, thereby enacting the use case scenarios defined in Tasks 2.1, 2.2, and 2.3 and documented in Deliverables D2.1, D2.2, and D2.3. The C++ helper class will be named EnvironmentKnowledgeBase.

In addition to functional code, the deliverable will include a report with sections detailing the specification, design, and implementation of the C++ helper class.

The population of the knowledge base, i.e., a C++ object instantiation of the EnvironmentKnowledgeBase class, is determined by the contents of a configuration file that contain a list of key-value pairs. One key-value pair will specify the filename of the file in which the knowledge is stored. One key-value pair will specify whether diagnostic data is to be printed to the terminal (e.g., verboseMode true). Others will be added, as required. The configuration file will be named environmentKnowledgeBaseConfiguration.ini.

The module design section will specify selected class data structures used to represent and store the knowledge read from the environment knowledge base file.

Code will be written and documented in adherence to the standards set out in Deliverable D3.2 Software Engineering Standards Manual.

A unit test will be provided to verify that the class meets the requirements and adheres to the specifications. It will do this by instantiating a C++ helper class object, and using the access methods to retrieve and write the contents of the environment knowledge base file to the terminal.

Description of Deliverables (continued)

D5.4.3 Robot Mission Interpreter
Deliverable type: software and report
This deliverable represents the outcome of Task 5.4.3. It comprises the documented software required to build and launch a module implemented as a single ROS node that interprets the scenario robot mission language developed in Task 5.4.2 and implements the specification of the two use case scenarios in that language, as documented in Deliverable D5.4.2. In does so by recruiting the robot and visitor behaviors documented in Deliverables D2.2 and D2.3, and the cultural knowledge encapsulated in Deliverable D5.4.1, and realized in the robot sensing modules developed in Work Package 4 and the robot behavior modules developed in Work Package 5. The ROS node will be named behaviorController.

The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

The operation of the behaviorController node is determined by the contents of a configuration file that contain a list of key-value pairs. One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true).

Note that this module does not support the simulator because it is dependent on input from person detection, robot localization, and speech event, all of which do not support the simulator; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D5.5.1.1 Gesture Execution
Deliverable type: software and report
This deliverable represents the first of the two outcomes of Task 5.5.1. It comprises the documented software required to build and launch a module implemented as a single ROS node that provides the robot with the ability to the Pepper robot to execute body and hand gestures. Hand gestures will include deictic, symbolic, and iconic non-verbal gestures. Body gestures will restricted to bowing and nodding, i.e., lowering and raising gaze. The ROS node will be named gestureExecution.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in the task description, identifying gaps and misalignments with actual needs.

The module specification section will define the functional characteristics, as follows. The module will execute five forms of gesture: deictic, symbolic, and iconic non-verbal hand gestures and bowing and nodding body gestures. The specifications for these gestures will be in joint space, except for deictic gestures which will be in Cartesian space. Some gestures, e.g., iconic and symbolic hand gestures will be specified by learning the required motions either by manual teleoperation, recording the joint angles, or by demonstration, using an RGB-D depth camera to determine the joint angles of human gestures in a skeletal model and mapping these to the robot joints. Other gestures, i.e., deictic hand gestures and body gestures, will be specified by gesture parameters, such as the pointing location for deictic gestures and the degree of inclination for bowing and nodding, and the joint angles will be computed using the kinematic model of the robot head, torso, and arms. For deictic gestures, which require the robot to point at objects in its environment, the pose of the robot in the world frame of reference will also be used.

If the arm cannot achieve the required pose for a deictic gesture, the robot should rotate to make the pose achievable, returning to the original orientation once the gesture is complete. Also, the arm should return to a neutral position by the robot's side when the gesture is complete.

Iconic and symbolic gestures will be defined by descriptors that specify the final gesture joint configuration and the manner in which that configuration is achieved. Descriptors comprise four elements. Each element is a key-value pair, where the value can be an identifier, a number, a vector of numbers, or a vector of a vector of numbers. The first key-value pair will specify the gesture type (e.g., type iconic | type symbolic). The second key-value pair will identify the ID number (e.g., ID 01). The fourth element is a vector of joint angles vectors. The number of joint angle vectors is equal to the number of way points, including the start joint configuration and the final gesture configuration. Body gestures have three joints: knee pitch, hip pitch, hip hip roll. Iconic and symbolic gestures have five joints: shoulder pitch, shoulder roll, elbow yaw, elbow roll, and wrist yaw.

Description of Deliverables (continued)

Before beginning the gesture, the arm is moved from its current joint configuration to the start joint configuration, i.e., the joint angles specified in the first vector in the vector of vector of joint angles. The number of elements in the vector of joint angles is determined by the gesture type. Descriptors for each gesture will be stored in an external descriptor file.

If an iconic or symbolic gesture involves two arms, they will be treated as a composite of two individual gestures, one for each arm.

Two approaches will be investigated to determine the joint angles for iconic and symbolic hand gestures. The first will involve the manual adjustment and recording of each joint by an operator. This will be done for each waypoint in the gesture. The second will involve learning the joint angles by demonstration, using an RGB-D camera and body tracking software with a skeleton model. In both cases, these angles will be in the gesture descriptor file and read from the file at run time.

The joint angles for bow and nod body gestures, as well as hand deictic gestures, will be computed at run time using the kinematic model of the robot and the bow angle, nod angle, or the location in the environment to which the robot should point. The bow angle, nod angle, and pointing location will be provided as an input to the module, along with the time in milliseconds that should elapse between the start of the gesture and the end of the gesture. The pointing location with respect to the robot body, specified by the shoulder pitch and shoulder roll angles, will be computed from the pointing location in the world frame of reference (and supplied as an input to the module) and the pose of the robot in the world frame of reference (provided by the module in Deliverable 4.2.4). No waypoints are required for deictic gestures; the joints will be actuated to achieve the target joint angles, interpolating linearly or adjusting the joint angles, joint angular velocities, and joint accelerations to mimic biological movement by using a kinematic model of biological motion (e.g., a power law or a minimum jerk law). It is assumed that the knee pitch angle is fixed during a bow body gesture and that the bow angle corresponds to the change in the hip pitch angle with respect to the default hip pitch angle. Similarly, it is assumed that the nod angle is the change in the head pitch angle with respect to the default head pitch angle. Finally, it is assumed that the arm and fingers are straight in a deictic gesture, with fixed values of elbow yaw, elbow roll, wrist yaw, and hand angles, so that the palm of the hand is directed upwards.

The input to the module will be a record comprising the gesture type (e.g., iconic | symbolic | deictic | bow | nod), the gesture ID for symbolic or iconic gestures (e.g., 01), the duration of the gesture in milliseconds, and either a bow angle in degrees (for a bow body gesture), or a nod angle in degrees (for a nod body gesture), or the three dimensional coordinates of a pointing location (for a deictic gesture). For deictic gestures, the module will also input the current robot pose from the robotLocalization module. The record input will be provided by a ROS service /gestureExecution/perform_gesture. The pose input will be provided by subscribing to a topic /robotLocalization/pose.

The output will be a sequence of joint angles, joint angular velocities, and, optionally, joint angular accelerations. Data will be published on the appropriate topics, as required; a ROS action will be used if necessary instead of the service.

The names of the topics to be used for each actuator will be read from a data file comprising a sequence of key-value pairs. The key is the name of the actuator. The value is the topic name.

Description of Deliverables (continued)

The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal. The operation of the gestureExecution module is determined by the contents of a configuration file that contains a list of key-value pairs. One key-value pair will specify the interpolation type. This indicates how the joint angles that define the trajectory in joint space between the current joint angles and the gesture joint angles are to be computed for body gesture and hand deictic gestures, and between way points for iconic and symbolic gestures. At least two options will be considered: (a) independent linear interpolation of each joint angle, and (b) biological motion, selecting the sequence of joint angular velocities and joint accelerations to form a trajectory in time and joint space that mimics biological movement (e.g., interpolation linear | interpolation biological). One key-value pair will specify the filename of the file in which the gesture descriptors are stored (e.g., gestureDescriptors gestureDescriptors.dat). One key-value pair will specify the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true). The configuration file will be named gestureExecutionConfiguration.ini.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through files, or ROS subscribers, services, or actions. Based on the module specification in the task description, it is planned to develop a driver to generate test input data and publish it on the topic to which the gestureExecution module subscribes for command input. It is planned that the module will use the physical robot to sink the control data published on the various actuator topics.

The module design section will specify selected message data structures used to represent and store the topic messages required for command input and to control the actuators, and the algorithms required to change frames of reference and compute the required joint angles.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will present the unit tests to verify that the module meets the module specifications. A unit test will be provided for each of the two test platforms: driver-stub unit test and physical robot unit test. The unit tests will be invoked by a launch file that brings up the required resources, e.g., any software to communicate with the physical robot. A test report will document the results of these tests.

Finally, a user manual will be included in the deliverable, explaining how to invoke, provide input to, and configure the gestureExecution module.

Note that this module does not support the simulator because it is dependent on input from robot localization which does not support the simulator; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D5.5.1.2 Programming by Demonstration
Deliverable type: software and report
This deliverable represents the second of the two outcomes of Task 5.5.1. It documents the development and functionality of a ROS package — the PepperTrace Programming by Demonstration Tool — designed to enable the Pepper robot to execute upper body and hand gestures learned through Programming by Demonstration (PbD). This software module is tailored for deployment on the real Pepper robot and is not intended for use in simulation.

The deliverable will include the following.

- The documented code for the `programming_by_demonstration` ROS package.
- A comprehensive report detailing the system architecture, component requirements, development procedures, and an explicit definition of the module's functional characteristics.
- Instructions for developers who wish to adapt the package for custom applications.
- A user manual to assist users in configuring and launching the module.

The package's interface design covers input, output, and control data, while also specifying appropriate data structures. All development activities will adhere to the software engineering standards outlined in Deliverable D3.2.

Note that this module does not support the simulator because it uses an external RealSense camera which is not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D5.5.2.1 English Text to Speech Conversion
Deliverable type: software and report
This deliverable represents the outcome of Task 5.5.2.1. It comprises the documented software function to convert English text to speech, specifically the text of the robot's interactions specified in Deliverable D2.2 Robot Behavior Specification and recruited in Deliverable D5.4.2 Robot Mission Language. The possibility of controlling loudness, pitch, and timbre of the speech will be investigated. The function will be named englishTTS.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the function. This will involve a review and update of the requirements as set out in this work plan, identifying gaps and misalignments with actual needs.

The function specification section will define the functional specification, detailing the input to output data transformation, expected input data, and expected output data. Functionally, this function will use a speech synthesis engine to convert English text to an audio file that can then be played on the robots loudspeakers.

The interface design section will include a specification of the data that are passed to the function, the data that are returned from the function. In addition, it will include the specification of a driver to source test input data, and a specification of stub to sink test output data. Based on the function specification, it is planned to develop a driver to generate test data based on parameters defined in the driver input file, using text defined in the Deliverable D2.2 Robot Behavior Specification. It is planned to develop a stub to sink the output data and play the audio file on a PC.

The module design section will specify selected algorithms and data structures, and develop a unit test plan. It is planned to investigate the feasibility of using the NAOqi audio utility ALTextToSpeech through a ROS interface.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will present the unit tests. At least two types of unit test will be delivered: (i) verification against module specification, and (ii) validation against requirements definition. Where feasible, an evaluation unit test against benchmark data or standard metrics will be delivered. A test report will document the results of these tests.

Note that this module does not support the simulator because the speakers are not available in Gazebo using the `\speech` topic; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D5.5.2.2 isiZulu Text to Speech Conversion
Deliverable type: software and report
This deliverable represents the outcome of Task 5.5.2.2. It comprises the documented software function to convert isiZulu text to speech, specifically the text of the robot's interactions specified in Deliverable D2.2 Robot Behavior Specification and recruited in Deliverable D5.4.2 Robot Mission Language. The possibility of controlling loudness, pitch, and timbre of the speech will be investigated. The function will be named isiZuluTTS.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the function. This will involve a review and update of the requirements as set out in this work plan, identifying gaps and misalignments with actual needs.

The function specification section will define the functional specification, detailing the input to output data transformation, expected input data, and expected output data. Functionally, this function will use a speech synthesis engine to convert English text to an audio file that can then be played on the robots loudspeakers.

The interface design section will include a specification of the data that are passed to the function, the data that are returned from the function. In addition, it will include the specification of a driver to source test input data, and a specification of stub to sink test output data. Based on the function specification, it is planned to develop a driver to generate test data based on parameters defined in the driver input file, using text defined in the Deliverable D2.2 Robot Behavior Specification. It is planned to develop a stub to sink the output data and play the audio file on a PC.

The module design section will specify selected algorithms and data structures, and develop a unit test plan. It is planned to investigate the feasibility of using the NAOqi audio utility ALTextToSpeech through a ROS interface.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will present the unit tests. At least two types of unit test will be delivered: (i) verification against module specification, and (ii) validation against requirements definition. Where feasible, an evaluation unit test against benchmark data or standard metrics will be delivered. A test report will document the results of these tests.

Note that this module does not support the simulator because the speakers are not available in Gazebo using the `\speech` topic; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D5.5.2.3 Kinyarwanda Text to Speech Conversion
Deliverable type: software and report
This deliverable represents the outcome of Task 5.5.2.3. It comprises the documented software function to convert Kinyarwanda text to speech, specifically the text of the robot's interactions specified in Deliverable D2.2 Robot Behavior Specification and recruited in Deliverable D5.4.2 Robot Mission Language. The possibility of controlling loudness, pitch, and timbre of the speech will be investigated. The function will be named kinyarwandaTTS.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the function. This will involve a review and update of the requirements as set out in this work plan, identifying gaps and misalignments with actual needs.

The function specification section will define the functional specification, detailing the input to output data transformation, expected input data, and expected output data. Functionally, this function will use a speech synthesis engine to convert English text to an audio file that can then be played on the robots loudspeakers.

The interface design section will include a specification of the data that are passed to the function, the data that are returned from the function. In addition, it will include the specification of a driver to source test input data, and a specification of stub to sink test output data. Based on the function specification, it is planned to develop a driver to generate test data based on parameters defined in the driver input file, using text defined in the Deliverable D2.2 Robot Behavior Specification. It is planned to develop a stub to sink the output data and play the audio file on a PC.

The module design section will specify selected algorithms and data structures, and develop a unit test plan. It is planned to investigate the feasibility of using the NAOqi audio utility ALTextToSpeech through a ROS interface.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will present the unit tests. At least two types of unit test will be delivered: (i) verification against module specification, and (ii) validation against requirements definition. Where feasible, an evaluation unit test against benchmark data or standard metrics will be delivered. A test report will document the results of these tests.

Note that this module does not support the simulator because the speakers are not available in Gazebo using the `\speech` topic; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D5.5.2.4 Integrated Text to Speech Conversion
Deliverable type: software and report
This deliverable represents the outcome of Task 5.5.2.4 to integrate the output of Tasks 5.5.2.1, 5.5.2.2, and 5.5.2.3. It comprises the documented software required to build and launch a module implemented as a single ROS node that converts English, isiZulu, and Kinyarwanda text to speech. The ROS node will be named textToSpeech.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in this work plan, identifying gaps and misalignments with actual needs.

The module specification section will define the functional specification, detailing the input to output data transformation, expected input data, expected output data, module configuration parameters, and a draft user manual. Functionally, this module will use a speech synthesis engine to convert English, isiZulu, or Kinyarwanda text to an audio file that can then be played on the robots loudspeakers. The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal. The operation of the module will be determined by parameters provided in a configuration file. This configuration file will contain a key-value pair to identify the language to be used (e.g., language English | language Kinyarwanda | language isiZulu). The configuration file will be named textToSpeechConfiguration.ini.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through ROS topics, services, or actions. In addition, it will include the specification of a driver to source test input data, and a specification of stub to sink test output data. Based on the module specification, it is planned to develop a driver to generate test data based on parameters defined in the driver input file, using text defined in the Deliverable D2.2 Robot Behavior Specification. It is planned to develop a stub to sink the output data and play the audio file on a PC.

The module design section will specify the required function calls to englishTTS, isiZuluTTS, and kinyarwandaTTS, and develop a unit test plan.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

Description of Deliverables (continued)

The unit testing section will present the unit tests. At least two types of unit test will be delivered: (i) verification against module specification, and (ii) validation against requirements definition. Where feasible, an evaluation unit test against benchmark data or standard metrics will be delivered. A test report will document the results of these tests. In addition, a user manual will be included in the deliverable, explaining how to invoke, provide input to, and configure the module.

Note that this module does not support the simulator because the speakers are not available in Gazebo using the `\speech` topic; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D5.5.3 Environment Map Generation
Deliverable type: software and report
This deliverable represents the outcome of Task 5.5.3. It comprises the documented software required to build and launch a module implemented as a single ROS node that generates metric workspace and configuration space maps of the environments that will be used in the two use case scenarios documented in Deliverable D2.1, augmented by the locations and identities of objects of interest, as specified in the use case scenarios. Thus the map will comprise both non-symbolic metric data and symbolic semantic data. These maps will be used by the Pepper robot when navigating and when making deictic gestures. The ROS node will be named mapGeneration.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in this work plan, identifying gaps and misalignments with actual needs.

The module specification section will define the functional specification, detailing the input to output data transformation, expected input data, expected output data, module configuration parameters, and a draft user manual. Functionally, this module will generate a workspace metric map, augmented by a list of the locations and identities of objects and regions of interest to which the robot can refer to and point to when interacting with humans. Such objects include walls, floor furniture, wall furniture, plants, steps (up and down), doors and doorways, signs, and elevators. This object-location data is to be generated interactively by displaying the workspace map in a window and allowing a user to use a cursor to identify a location and label it with a single-word identifier. These identifiers should correspond to the labels that are used in the use case scenario description in Deliverable D6.1 written using the robot mission language in Deliverable D5.4.2.

The input to the module will be a record comprising the filenames of the files to which the workspace and configuration space maps are to be written, and the data file in which the list of labelled objects of interest and their locations on the maps is to be stored.

The output will be a non-symbolic metric data map of the environment workspace, and a list of labelled objects of interest and their locations on the maps. The map should be in a format that can be rendered as an image. Data will be published on the appropriate topics. Image dilation will be used to convert the workspace map to a configuration space map using a structuring element that models the size and shape of the robot's base. This will provide a second output.

The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal.

Description of Deliverables (continued)

The operation of the module will be determined by parameters provided in a configuration file that contains a list of key-value pairs. One key-value pair will specify the platform on which the tests are to be run, i.e., the physical Pepper robot or the Pepper simulator (e.g., platform robot | platform simulator). One key-value pair will specify the map generation mode, i.e., using CAD data or SLAM (e.g., mode CAD | mode SLAM). One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true). The configuration file will be named mapGenerationConfiguration.ini.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through ROS topics, services, or actions. In addition, it will include the specification of a driver to source test input data, and a specification of stub to sink test output data. Based on the module specification, it is planned that the interface of this module will develop a driver to generate test data for CAD mode based on parameters defined in the driver input file, i.e., the geometric specification the geometry of the environment for each use case. It is planned to develop a stub to sink the output data and display the metric map image of the workspace using OpenCV, with the workspace image annotated with the object labels at the appropriate locations.

The module design section will specify selected algorithms and data structures, and develop a unit test plan. It is planned to adopt two strategies: (i) map generation from a priori data (i.e., environment CAD data) and (ii) map generation using Simultaneous Localization and Mapping (SLAM).

The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are described in Deliverable D5.5.3.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

The unit testing section will present the unit tests to verify that the module meets the module specifications. A unit test will be provided for each of the two test platforms: (i) physical robot unit test, and (ii) simulator unit test. The unit tests will be invoked by a launch file that brings up the required resources, e.g., any software to communicate the the physical robot, the simulator. For the physical robot or simulator unit tests to work correctly, the node configuration file (*.ini) has to have the correct platform selected. A test report will document the results of these tests.
In addition, a user manual will be included in the deliverable, explaining how to invoke, provide input to, and configure the module.

Note that this module does not support the simulator in SLAM mode because it uses external devices which are not available in Gazebo; see the notes on Version 3.0 in the Document History.

Description of Deliverables (continued)

D5.5.4 Robot Navigation
Deliverable type: software and report
This deliverable represents the outcome of Task 5.5.4. It comprises the documented software required to build and launch a module implemented as a single ROS node that controls the locomotion of the Pepper robot so that it navigates its environment, in which there are fixed inanimate obstacles and moveable obstacles in the form of humans, from its current position along the shortest path to a destination position and orientation identified in the use case scenario script. Navigation will be effected by identifying waypoints along the navigation path and the robot will move from waypoint to waypoint. The ROS node will be named robotNavigation.

In addition to functional code, the deliverable will include a report with sections detailing the outputs of each phase of the software development process.

The requirements definition section will specify the functional needs of users of the module. This will involve a review and update of the requirements as set out in this work plan, identifying gaps and misalignments with actual needs.

The module specification section will define the functional specification, detailing the input to output data transformation, expected input data, expected output data, module configuration parameters, and a draft user manual. Functionally, this module will optionally augment a metric workspace map of the robot's environment with obstacles corresponding to the location of any humans that have been detected in the robot's field of view. The size of the human obstacle will be determined using culturally sensitive proxemics. This augmented workspace map will then be used to generate a configuration space map that constrains the robot's path from its current location to its target location using either Dijkstra's algorithm and the A* algorithm. Waypoints will be identified using one of two candidate techniques: equidistant waypoints and high path curvature waypoints. Locomotion from waypoint to waypoint will be effected using one of two locomotion algorithms: Multiple Input Multiple Output (MIMO) and divide and conquer (DnQ).

The input to the module will be a record identifying the destination pose for the robot, specified by the x and y coordinates of the location and the direction the robot should face (i.e., the direction of the X-axis in the robot base frame, all specified in the workspace frame of reference). The module will also acquire the robot's current pose using the module in Deliverable D4.2.4 Robot Localization, the cultural knowledge regarding proxemics in a file produced in Deliverable D1.2 African Modes of Social Interaction, and the workspace map in a file produced in Deliverable D5.5.3.

The output will be a sequence of forward velocity and angular velocity values published on the cmd_vel topic. The configuration space map will also be rendered as an image. The planned path will be rendered graphically on this map. The module can also request the robotLocalization to reset the robot's pose using absolute pose estimation.

The names of the topics to be used for each actuator will be read from a data file comprising a sequence of key-value pairs. The key is the name of the actuator. The value is the topic name.

Description of Deliverables (continued)

The module can run in normal mode or verbose mode. In verbose mode, data that is published to topics is also printed to the terminal. Also, if the module is operating in verbose mode, the configuration space map image will be displayed in an OpenCV window.

The operation of the module will be determined by parameters provided in a configuration file containing a list of key-value pairs. One key-value pair will specify the filename of the file in which the workspace map is stored (e.g., map scenarioOneMap.dat). One key-value pair will specify the path planning algorithm to be used (e.g., pathPlanning Dijkstra | pathPlanning A*). One key-value pair indicates whether or not to take into consideration social constraints while navigating (e.g., socialDistance TRUE | socialDistance FALSE). One key-value pair will specify the filename of the file in which the physical Pepper robot sensor and actuator topic names are stored (e.g., robotTopics pepperTopics.dat). One key-value pair will specify whether diagnostic data is to be printed to the terminal and diagnostic images are to be displayed in OpenCV windows (e.g., verboseMode true). The configuration file will be named robotNavigationConfiguration.ini.

The interface design section will include a specification of the data that are input to the module, the data that are output from the module, and the data that are used to control the operation of the module, including the manner in which this data is made available or accessed, either through ROS topics, services, or actions. In addition, it will include the specification of a driver to source test input data, and a specification of stub to sink test output data. Based on the module specification, it is planned to develop a driver to generate test data, i.e., target robot poses for each use case scenario workspace map. It is planned to develop a stub to sink the output data and display the metric map images of the workspace and configuration space using OpenCV.

The module design section will specify selected algorithms and data structures, and develop a unit test plan. It is planned to adopt two path planning algorithms (Dijkstra's algorithm and the A* algorithm) and two waypoint selection algorithms (equidistant and high curvature waypoints).

The software development process will involve requirements definition, module specification, interface design, module design, coding, and unit testing. The anticipated outputs of each of these phases are described in Deliverable D5.5.3.

The coding section will contain the functional program code with internal documentation, any required build files, driver program code, and stub program code. Code will be written and documented in adherence to the standards to be developed in Deliverable D3.2 Software Engineering Standards Manual.

Description of Deliverables (continued)

The unit testing section will present a unit test to verify that the module meets the module specifications. A unit test will be provided for the physical robot. The unit test will be invoked by a launch file that brings up the required resources, e.g., any software to communicate with the physical robot. A test report will document the results of this test.

In addition, a user manual will be included in the deliverable, explaining how to invoke, provide input to, and configure the module.

Note that this module does not support the simulator because it is dependent on input from robot localization which does not support the simulator; see the notes on Version 3.0 in the Document History.

D5.6 Use Case Feedback
Deliverable type: report.
This deliverable represents the outcome of Task 5.6. It documents any adjustments that need to be made to the robot behaviors based on the evaluation of the two use cases in work package WP6.

2.4.6 Work Package 6: Use Case Demonstration and Evaluation

Objectives

1. Carry out initial demonstrations of the system architecture in the two use cases.
2. Evaluate the success of the demonstrations and identify any adjustments to the output from work packages WP1 - WP5.
3. Carry out final demonstrations.
4. Re-evaluate the success of the demonstrations.

Description of Work

Task 6.1: Use Case Implementation (M19–M21)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to implement both use cases using the outcomes of WP1 - WP5, i.e., the cultural knowledge, the scenario specification, and the integrated robot's sensory and interaction capabilities. This objective will be achieved by running the unit test and system tests in Deliverable D3.3. The outcome of this task is described in Deliverable D6.1.

Task 6.2: Use Case Evaluation (M22–M24)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to evaluate the implementation using the Robot Social Attribute Scale (RoSAS) and produce a set of required adjustments for the interaction primitives and design patterns. The task uses the outcome of Task 6.1 as documented in Deliverable D6.1. The outcome of this task is described in Deliverable D6.2.

Task 6.3: Use Case Re-Evaluation (M34–M36)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to re-evaluate the implementation after completing the adjustments in work packages WP1 – WP5 arising from the evaluation in Task 6.2. The outcome of this task is described in Deliverable D6.3.

List of Deliverables

Number	Deliverable Title	Responsible Partner	Delivery Month
D6.1	Use Case Implementation	Wits	M21
D6.2	Use Case Evaluation	Wits	M24
D6.3	Use Case Re-Evaluation	Wits	M36

Description of Deliverables

D6.1 Use Case Implementation
Deliverable type: demonstration and report
This deliverable represents the outcome of Task 6.1. It is a demonstration of the complete working system for the two use cases defined in Work Package WP2.

D6.2 Use Case Evaluation
Deliverable type: report
This deliverable represents the outcome of Task 6.2. It will evaluate and document the success of the initial use case implementation using the Robot Social Attribute Scale (RoSAS).

D6.3 Use Case Re-Evaluation
Deliverable type: report
This deliverable represents the outcome of Task 6.3. It will document the success of the final use case implementation using the Robot Social Attribute Scale (RoSAS).

2.4.7 Work Package 7: Dissemination and Impact

Objectives

1. Create a project website.
2. Disseminate the ongoing status of the project in several forums, including the project website.
3. Publish research results in conferences and journals.
4. Make the project software and data freely available online.
5. Organize a summer school.

Description of Work

Task 7.1: Online Presence (M1–M36)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to communicate project aims and results to the general public and stakeholders. This objective will be achieved by creating a project website to describe the motivation for and goals of the project, to periodically post updates on progress, and highlight results, including publications and forthcoming events. The outcome of this task is described in Deliverable D7.1.

Task 7.2: Dissemination Activities (M1–M36)
Responsible Partner: University of the Witwatersrand.
The objective of this task is effect scientific dissemination through leading international conferences and journals, as well as through forums such as AI Saturdays, Knowledge for All (K4All), and AI4D. A record of all communication actions will be kept and incorporated in a yearly dissemination and communication report. The outcome of this task is described in Deliverable D7.2.

Task 7.3: Open-Source Software Repository (M5–M36)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to make software, data, and protocols publicly available on GitHub and maintain the repositories. The outcome of this task is described in Deliverable D7.3.

Task 7.4: Summer School (M34–M36)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to organize a summer school on culturally-sensitive human-robot interaction in 2026. The outcome of this task is described in Deliverable D7.4

List of Deliverables

Number	Deliverable Title	Responsible Partner	Delivery Month
D7.1	Online Presence	CMU-Africa	M3
D7.2	Dissemination Activities	Wits	M12, M24, M36
D7.3	Open-Source Software Repository	CMU-Africa	M5
D7.4	Summer School	Wits	M34

Description of Deliverables

D7.1 Online Presence
Deliverable type: website, wiki, and report
This deliverable represents the outcome of Task 7.1. It is an operational website detailing the goals, status, and results of the project. It will be delivered at month 3 and will be maintained throughout the duration of the project.

D7.2 Dissemination Activities
Deliverable type: report
This deliverable represents the outcome of Task 7.2. It documents the reports, papers, articles, presentations, and demonstrations that have been carried out. Three versions of the deliverable will be produced, one at the end of each year.

D7.3 Open-Source Software, Data, and Designs
Deliverable type: online repository and report
This deliverable represents the outcome of Task 7.3. It is a GitHub software repository of the software and data (i.e., the cutural knowledge) developed in the project. The initial repository will be created by month 5 and it will be updated as software is developed throughout the duration of the project.

D7.4 Summer School
Deliverable type: event and report
This deliverable represents the outcome of Task 7.4. It will be organized at the end of the project, to enable the wider community to use the results of the project.

2.4.8 Work Package 8: Project Management

Objectives

1. Carry out coordination, finance, and management activities.
2. Prepare and sign a consortium agreement.
3. Prepare and execute a gender action plan.

Description of Work

Task 8.1: Project Coordination (M1–M36)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to coordinate the technical work in the project. Progress in each work package and each task will be monitored weekly to identify any potential issues and to take the appropriate timely corrective actions. This task will involve the organization of in-person project meetings every three months and writing six-monthly project progress reports, including a final report setting out the project activities, significant accomplishments, and reference to published papers. For day-to-day communication and coordination, we will set up a Discord platform dedicated to the projects, with individual channels for each work package and each task. The outcome of this task is described in Deliverable D8.1.

Task 8.2: Administration (M1–M36)
Responsible Partner: Carnegie Mellon University Africa.
The objective of this task is to plan and monitor financial expenditure, and submit periodic financial reports. The outcome of this task is described in Deliverable D8.2.

Task 8.3: Risk Management (M1–M36)
Responsible Partner: University of the Witwatersrand.
Initial risks and appropriate mitigation strategies have already been identified in Table 4. The objective of this task is to focus on monitoring these risks as the project progresses, and take appropriate corrective actions, where necessary. It also monitors the possible emergence of new risks and updates the risk plan accordingly. Any updates will be reported in periodic reports. The outcome of this task is described in Deliverable D8.3.

Task 8.4: Consortium Agreement (M1–M4)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to draft and agree a consortium agreement under which the rights and obligations of the partners are set out, with particular attention being paid to intellectual property rights, and the adoption of an appropriate open source licence to be adopted for all project deliverables. The outcome of this task is described in Deliverable D8.4.

Description of Work (continued)

Task 8.5: Gender Action Plan (M1–M4)
Responsible Partner: University of the Witwatersrand.
The objective of this task is to draft a gender action plan with the specific intent of ensuring there is no gender bias in any activity directed at stakeholders, i.e., in the ethnographic studies and in the execution and evaluation of the two use cases. The outcome of this task is described in Deliverable D8.5.

List of Deliverables

Number	Deliverable Title	Responsible Partner	Delivery Month
D8.1	Progress Report	CMU-Africa	M6, M12, ..., M36
D8.2	Expenditure Report	CMU-Africa	M6, M12, ..., M36
D8.3	Risk Assessment	Wits	M12, M24, M36
D8.4	Consortium Agreement	Wits	M4
D8.5	Gender Action Plan	Wits	M4

Description of Deliverables

D8.1 Progress Report
Deliverable type: report
This deliverable documents progress in each work package and in each task. It is produced every six months. A final report will be produced in month 36 at the end of the project. This report will set out the project activities, significant accomplishments, and reference to published papers.

D8.2 Expenditure Report
Deliverable type: report
This deliverable represents the outcome of Task 8.2. It summarizes the financial standing of the project under each budget heading. It is produced every six months.

D8.3 Risk Assessment
Deliverable type: report
This deliverable represents the outcome of Task 8.3. It documents any risk mitigation actions that have been taken. It is produced at the end of each year.

D8.4 Consortium Agreement
Deliverable type: report
This deliverable represents the outcome of Task 8.4. It sets out the rights and obligations of the partners regarding intellectual property rights. It also identifies the open source licence that is adopted for all project deliverables.

D8.5 Gender Action Plan
Deliverable type: report
This deliverable represents the outcome of Task 8.5. It documents the actions that are to be taken to ensure that there is no gender bias in any activity directed at stakeholders, i.e., in the ethnographic studies and in the execution and evaluation of the two use cases.

Table 3: List and Schedule of Milestones

Number	Milestone Title	Related WP No.	Month	Means of Verification
M1	Completion of study to acquire African cultural knowledge	WP1	6	D1.1
M2	System architecture complete	WP3	9	D3.1
M3	Software implementation phase 1 complete	WP3, WP4, WP5	18	D3.3
M4	Use cases implementation and evaluation phase 1 complete	WP6	24	D6.2
M5	Software implementation phase 2 complete	WP4, WP5	33	D3.3
M6	Use cases implementation and evaluation phase 2 complete	WP6	36	D6.3

Table 4: Critical risks for implementation

Description of Risk	WP Involved	Risk Mitigation Strategy
African cultural knowledge are not effective in use cases	WP3, WP6	T6.2 produces a set of required adjustments which are then implemented in T1.4.
Implementation of the system architecture for use cases is insufficient	WP3	T6.2 produces a set of required adjustments which are then implemented in T3.5
Robot sensing and analysis does not perform adequately	WP4	T6.2 produces a set of required adjustments which are then implemented in T4.4
Robot behaviors do not perform adequately	WP5	T6.2 produces a set of required adjustments which are then implemented in T5.6.
T6.4 Use case evaluation does not achieve sufficiently high user ratings in the evaluation	WP1–WP6	T6.2 identifies adjustments; these are implemented in T1.4, T2.4, T3.5, T4.4, and T5.6.

Table 5: Summary of effort

Partner	WP1	WP2	WP3	WP4	WP5	WP6	WP7	WP8	Total
Carnegie Mellon University Africa	3.02	1.89	1.89	7.05	4.02	3.35	1.52	1.25	24.00
University of the Witswatersrand	2.62	1.68	1.68	6.45	3.71	2.28	1.23	0.06	20.23
Total	5.64	3.57	3.57	13.50	7.73	6.15	2.75	1.31	44.23

References

Alupo, C. D., Omeiza, D., & Vernon, D. (2022). Realizing the potential of AI in Africa. In M. I. A. Ferreira & O. Tokhi (Eds.), *Towards trustworthy artificial intelligence systems.* Springer.

Bartneck, C., Belpaeme, T., Eyssel, F., Kanda, T., Keijsers, M., & Sabanovic, S. (2020). *Human-robot interaction – an introduction.* Cambridge University Press.

Bartneck, C., Nomura, T., Kanda, T., Suzuki, T., & Kennsuke, K. (2005). Cultural differences in attitudes towards robots. In *Proceedings of the aisb symposium on robot companions: Hard problems and open challenges in human-robot interaction* (pp. 1–4).

Bruce, N. D. B., & Tsotsos, J. K. (2009). Saliency, attention, and visual search: An information theoretic approach. *Journal of Vision*, *9*(3), 1–24.

Brugali, D., & Scandurra, P. (2009, December). Component-Based Robotic Engineering (Part I). *IEEE Robotics and Automation Magazine*, 84–96.

Brugali, D., & Shakhimardanov, A. (2010, March). Component-Based Robotic Engineering (Part II). *IEEE Robotics and Automation Magazine*, 100–112.

Bruno, B., Chong, N. Y., Kamide, H., Kanoria, S., Lee, J., Lim, Y., ... Pecora, F. (2017a). The CARESSES EU-Japan project: Making assistive robots culturally competent. In *arXiv 1708.06276.*

Bruno, B., Chong, N. Y., Kamide, H., Kanoria, S., Lee, J., Lim, Y., ... Sgorbissa, A. (2017b). Paving the way for culturally competent robots: A position paper. In *26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)* (pp. 553–560). Lisbon, Portugal.

Bruno, B., Recchiuto, C. T., Papadopoulos, I., Saffiotti, A., Koulouglioti, C., Menicatti, R., ... Sgorbissa, A. (2019). Knowledge representation for culturally competent personal robots: requirements, design principles, implementation, and assessment. *International Journal of Social Robotics*, *11*(3), 515–538.

Carpinella, C. M., Wyman, A. B., Perez, M. A., & Stroessner, S. J. (2017). The robotic social attributes scale (RoSAS): Development and validation. In *Proceedings of the 2017 ACM/IEEE international conference on human-robot interaction* (pp. 254–262). New York, NY, USA: Association for Computing Machinery.

Dignum, V. (2023). Responsible artificial intelligence: Recommendations and lessons learned. In D. O. Eke, K. Wakunuma, & S. Akintoye (Eds.), *Responsible AI in Africa – challenges and opportunities.* Palgrave Macmillan Cham.

Eke, D. O., Wakunuma, K., & Akintoye, S. (Eds.). (2023). *Responsible AI in Africa – challenges and opportunities.* Palgrave Macmillan Cham.

euROBIN. (2023). *The European Network of Excellence in robotics.* Retrieved from https://www.eurobin-project.eu/

Heineman, G. T., & Council, W. T. (2001). *Component-Based Software Egineering: Putting the pieces Together.* Reading, Massachusetts: Addison-Wesley.

Itti, L., & Koch, C. (2000). A saliency-based search mechanism for overt and covert shifts of visual attention. *Vision Research*, *40*, 1489–1506.

Itti, L., & Koch, C. (2001). Computational modelling of visual attention. *Nature Reviews Neuroscience*, *2*, 194–203.

Itti, L., Koch, C., & Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*, 1254ñ-1259.

Kahn, P. H., Freier, N. G., Kanda, T., Ishiguro, H., Ruckert, J. H., Severson, R. L., & Kane, S. K. (2008). Design patterns for sociality in human-robot interaction. In *Proc. 3rd ACM/IEEE international conference on human-robot interaction* (pp. 97 – 104).

Kaplan, F. (2004). Who is afraid of the humanoid? investigating cultural differences in the acceptance of robots. *International Journal of Humanoid Robotics*, *1*(3), 1 – 16.

Khaliq, A., Köckemann, U., Pecora, F., Saffiotti, A., Bruno, B., Recchiuto, C., ... Chong, N. (2018). Culturally aware planning and execution of robot actions. In *IEEE/RSJ international conference on intelligent robots and systems (IROS)* (pp. 326–332). Madrid, Spain.

Lee, J. D., & See, K. A. (2004). Trust in automation: Designing for appropriate reliance. *Human Factors*, *46*(1), 50–80.

Lim, V., Rooksby, M., & Cross, E. S. (2021). Social robots on a global stage: Establishing a role for culture during human–robot interaction. *International Journal of Social Robotics*, *13*, 1307–1333.

Olasunkanmi, A. (2011). Development in Africa: The need for a culture-sensitive approach. *Journal of Sociology and Social Anthropology*, *2*(2), 97–101.

Rea, F., Metta, G., & Bartolozzi, C. (2013, December). Event-driven visual attention for the humanoid robot iCub. *Frontiers in Neuroscience*, *7*(Article 234), 1–11.

Research and Markets. (2022). *Global Social Robots Market.* Retrieved from https://www.researchandmarkets.com/reports/5120156

Rose, J. (2010). *Software innovation: eight work-style heuristics for creative software developers.* Software Innovation, Dept. of Computer Science, Aalborg University.

Sciutti, A., Mara, M., Tagliasco, V., & Sandini, G. (2018). Humanizing human-robot interaction: On the importance of mutual understanding. *IEEE Technology and Society Magazine*, *37*(1), 22–29.

Szyperski, C. (2002). *Component software: Beyond object-oriented programming.* Reading, Massachusetts: Addison-Wesley.

Tsotsos, J. K. (2006). Cognitive vision need attention to link sensing with recognition. In H. I. Christensen & H.-H. Nagel (Eds.), *Cognitive vision systems: Sampling the spectrum of approaches* (Vol. 3948, pp. 25–36). Heidelberg: Springer.

Tsotsos, J. K. (2011). *A computational perspective on visual attention.* Cambridge MA: MIT Press.

Tsotsos, J. K., Culhane, S., Wai, W., Lai, Y., David, N., & Nuflo, F. (1995). Modeling visual attention via selective tuning. *Artificial Intelligence*, *78*, 507–547.

Vernon, D., Billing, E., Hemeren, P., Thill, S., & Ziemke, T. (2015). An architecture-oriented approach to system integration in collaborative robotics research projects - an experience report. *Journal of Software Engineering for Robotics*, *6*(1), 15–32.

Young, J. E., Sung, J., Voida, A., Sharlin, E., Igarashi, T., Christensen, H. I., & Grinter, R. E. (2011). Evaluating human-robot interaction: Focusing on the holistic interaction experience. *International Journal of Social Robotics*, *3*(1), 53–67.

Zaharescu, A., Rothenstein, A. L., & Tsotsos, J. K. (2004). Towards a biologically plausible active visual search model. In L. Paletta, J. K. Tsotsos, E. Rome, & G. Humphreys (Eds.), *Proceedings of the second international workshop on attention and performance in computational vision, WAPCV* (Vol. LNCS 3368, pp. 133–147). Berlin: Springer.

Document History

Version 1.0

First draft.
David Vernon.
8 June 2023.

Version 1.1

Work packages WP4 and WP5 partially populated with task and deliverable descriptions.
Typographical errors fixed throughout the document.
David Vernon.
9 June 2023.

Version 1.2

Implemented several changes to address minor issues that arose as a consequence of the modifications introduced when producing Version 1.1.
David Vernon.
13 June 2023.

Version 1.3

Reviewed WP4 and WP5, made changes and populated some tasks and deliverable descriptions.
David Vernon.
27 June 2023.

Version 1.4

Updated appearance of cover page to be more consistent with the style of the deliverables.
David Vernon.
30 June 2023.

Version 1.5

Set start date to July 1, 2023 to align with first six-monthly progress report in December 31, 2023, and final report in June 30, 2026, as specified in the grant agreement.
Added lead partner and participating partner to each task.
Revised and expanded task descriptions and deliverable descriptions in Work Packages 1 - 8 to adhere to a common format and level of detail.
David Vernon.
17 July 2023.

Version 1.6

Fixed several typographical errors.
David Vernon.
26 July 2023.

Version 1.7

Fixed more typographical errors.
Changed delivery date of D7.1 from month 6 to month 4.
David Vernon.
28 July 2023.

Version 1.8

Rationalized the scheduling of all tasks and updated the Gantt chart accordingly.
David Vernon.
1 August 2023.

Version 1.9

Added a wiki to D7.1 Online Presence. Added report type to D7.1 and D7.3 Open-source Software Repository, so that there is now a report to document the creation of the website, the wiki, and the software repository. Brought forward the launch of the website and wiki to month 3 and the launch of the software repository to month 5, when the first software is due to be delivered.
David Vernon.
3 August 2023.

Version 1.10

Changed Lead Partner to Responsible Partner and removed Participating Partner from the task descriptions to comply with the requirement that activities in the Statement of Work (SoW), i.e., tasks in the work plan, can be assigned to only one partner.
Provisionally assigned responsibilities for the following tasks and subtasks to CMU-Africa: 2.1, 2.2, 2.3, 3.1, 3.2, 3.3, 3.4, 4.1, 4.2.1, 4.2.2, 4.2.3, 4.3.2, 5.1, 5.2, 5.3, 5.5.1, 5.5.3, 5.5.4, 7.1, 7.3, 8.1, 8.2.
Added missing task descriptions for Tasks 4.4 and 5.6, and deliverable descriptions for Deliverables D4.4 and D5.6.
David Vernon.
6 August 2023.

Version 1.11

Assigned responsibilities for Task 4.2.4 for the following tasks and subtasks to CMU-Africa.
David Vernon.
11 August 2023.

Version 2.0

Assigned responsibilities for remaining tasks to CMU-Africa and the University of the Witwatersrand to reflect two-thirds effort for CMU-Africa and one-third for Wits on the basis of the allocated budget.
David Vernon.
15 August 2023.

Version 2.1

Updated the specification of Deliverables D4.1 and D5.1: using key-value pairs in the input files, using .dat file extension for data files, and .ini file extension of configuration files.
David Vernon.
16 August 2023.

Version 2.2

Updated the description of Task 5.5.1 and specification of Deliverable D5.5.1 to reflect the need to specify deictic gestures in Cartesian space, i.e., by specifying the pointing location in a world frame of reference.
David Vernon.
19 August 2023.

Version 2.3

Updated the description of Task 5.5.2: split the task into four, one each for English, isiZulu, and Kinyarwanda text-to-speech, and one for integrating the three, with corresponding deliverables D5.5.2.1 – D5.5.2.4.
Changed file extensions for data files from .txt to .dat. Changed file extensions for configuration files from .txt to .ini.
David Vernon.
29 August 2023.

Version 2.4

Sensor software deliverables (D4.2.1, D4.2.2, D4.2.3, D4.2.4, D4.2.5) now have three unit tests: (i) driver-stub test, (ii) physical robot test, and (iii) simulator test (if feasible).

Actuator software deliverables (D5.2, D5.3, D5.5.1, D5.5.2, D5.5.3) now have two unit tests: (i) physical robot test, and (ii) simulator test.

The unit tests will be invoked by a launch file that brings up the required resources, e.g., any software to communicate the the physical robot, the simulator, or the driver and stub.

The driver of a sensor software deliverable will publish the sensor data on both the physical robot sensor topics and the simulator sensor topics.

For the physical robot or simulator unit tests to work correctly, the node configuration file (*.ini) has to have the correct platform selected.

D4.2.1: added a key-value pair to the configuration file to specify whether to use the physical robot or the simulator.

Updated Gantt chart.

All changes are in red font.They will revert to black font in the next version.

David Vernon.
28 September 2023.

Version 2.5

Updated Deliverable D4.3.1 Tablet PC Event to give the ROS node a name: tabletEvent.
Updated Deliverable D4.3.2 Speech Event to give the ROS node a name: speechEvent.
Updated Deliverable D5.4.3 Robot Mission Interpreter to give the ROS node a name: scriptInterpreter.
David Vernon.
23 October 2023.

Version 2.6

Updated Deliverable D5.4.1 to specify explicitly that it produces a software module, i.e., a ROS node: knowledgeBase.
David Vernon.
25 October 2023.

Version 2.7

Updated Deliverable D5.1 Actuator Tests to use an average angular velocity, thereby allowing the joint actuators to be controlled by specifying the duration of the rotation.
David Vernon.
27 October 2023.

Version 2.8

Updated Deliverable D4.1 Sensor Tests to remove the requirement for either a driver or a stub. A driver makes no sense as the tests are for the physical robot and the simulator. A stub isn't required since the node can write the data to the terminal or to an OpenCV image display.
David Vernon.
16 November 2023.

Version 2.9

Updated Deliverables D4.2.1 and D4.2.2 to include the label in the record containing the output data.
David Vernon.
24 November 2023.

Version 2.10

Updated Deliverables D4.1 to remove mention of the stub code.
David Vernon.
28 November 2023.

Version 2.11

Updated progress on Gantt chart
David Vernon.
14 December 2023.

Version 2.12

Updated description of Task 4.2.2 and Deliverable 4.2.2 to include two new configuration parameters that allow a false reject tolerance and spatial tolerance when assigning the same face label.

Updated description of Task 5.2 and Deliverable 5.2 to remove the mention of proxy actuator topics and allow instead the Interaction Manager subsystem to start and stop the generation of animate behaviour.

Revised description of Task 2.2 and Deliverable D2.2 to reflect new objective of identifying the ROS nodes that will implement the robot action capabilities.

Revised description of Task 2.3 and Deliverable D2.3 to reflect new objective of identifying the ROS nodes that will implement the robot perception capabilities.

Renamed integratedTTSConversion to integratedTTS, isiZuluTTSConversion to isiZuluTTS, englishTTSConversion to englishTTS, kinyarwandaTTSConversion to kinyarwandaTTS. Changes are set in red font to make them easier to see.

David Vernon.
31 December 2023.

Version 2.13

Updated description of Deliverable 4.2.1 to include two new configuration parameters that allow a false reject tolerance and spatial tolerance when assigning the same face label.

Changed specification of Task 4.3.1 and D4.3.1 Tablet PC event. Instead of implementing a finite state machine as originally planned, this node now simply provides a service that receives a message to be displayed on the tablet PC and responds with the visitor's response. The finite state machine, which effectively determines the robot behaviour as a function of interactions with the visitor, is specified in the use case scenario script (Task 5.4.2) and implemented by the use case robot mission interpreter (Task 5.4.3).

Updated description of Deliverable 5.2 to allow operation to be started and stopped by the Interaction Manager subsystem.

Updated description of Deliverable 5.3 to allow for the new stereo camera on the Pepper robot.

Added to the specification of Deliverable 5.5.1 to required the robot to rotate if the arm cannot achieve the required pose for a deictic gesture, in order to make the pose achievable, returning to the original orientation once the gesture is complete. Also, in all cases, specified that the arm should return to a neutral position by the robot's side when the gesture is complete.

Rationalized ROS topic names to adhere to the recommended convention at https://wiki.ros.org/ROS/Patterns/Conventions, following common C variable naming conventions: lower case, with underscore separators. We also prepend the node name and a "/" so that it is explict which node publishes the topic. The recommendations do not provide a convention for the node name so we have adopted camelCase for node names, throughout. The topic affected are as follows. /saliencyImage → /overtAttention/saliency_image.
/personDetectionData → /personDetection/data.
/personDetectionRGBImage → /personDetection/rgb_image.
/faceDetectionData → /faceDetection/data.
/faceDetectionImage → /faceDetection/image.
/soundDetectionData → /soundDetection/data.
/speechEventText → /speechEvent/text.

Rationalized values in key-value pairs to use the values that are used the pepperTopics.dat and simulatorTopics.dat files: FrontCamera, BottomCamera, and DepthCamera, instead of topCamera, bottomCamera, and depthCamera; also introduced StereoCamera.

Removed bars from Gantt chart for tasks that have subtasks.

Changes are set in red font to make them easier to see.

David Vernon.
9 January 2024.

Version 2.14

Rationalized the plans for text-to-speech. Instead of four TTS nodes — English, Kinyarwanda, isiZulu, and an integration version — these is now just one textToSpeech node which calls three functions. Thus Deliverables D5.5.2.1 - D5.5.2.3 now produce functions, not nodes.

Added functionality to Task 5.3 and Deliverable D5.3 Attention Subsystem to gaze at a given location in the environment.

Changes are set in red font to make them easier to see.

David Vernon.
12 January 2024.

Version 2.15

Rationalized the plans for robot localization, Task 4.2.4 and Deliverable D4.2.4. The possibility of triangulation using two landmarks is removed because the distance to the landmark from the robot cannot be adequately determined since the robot's range sensors are not sufficiently accurate. Additional functionality is added to allow the module to service requests to reset the robot's pose using absolute pose estimation.

Removed the need to specify in Task 5.5.4 and Deliverable D5.5.4 the filename of the file in which the culturally sensitive social knowledge is stored. This knowledge will be generated by calling a service provided by the knowledgeBase node.

Changed "System Architecture Design" to "System Architecture", throughout.

Changes are set in red font to make them easier to see.

David Vernon.
21 January 2024.

Version 2.16

Diagnostic images are no longer published on topics. Instead, each module can operation in two modes: normal and verbose. In verbose mode, data that is published to topics is also printed to the terminal. Also, if the module is operating in verbose mode, the any images will be displayed in an OpenCV window. This removes the need for a GUI module and allows the diagnostic output of each module to be customized by specifying the mode of operation using a configuration file key-value pair `verboseMode true` or `verboseMode false`. These changes are set in blue font to make them easier to see.

David Vernon.
24 January 2024.

Version 2.17

Updated to address a number of issues, as follows.
Changed `menuData.dat` to `tabletEventInput.dat`.
Changed Task 5.5.2.3 isiZulu ... to Task 5.5.2.2 isizulu ... and fixed "English" typo.
Fixed typos in Task 5.5.2.3 Kinyrwanda ...
Fixed typos in Task 5.5.2.
David Vernon.
25 January 2024.

Version 2.18

Added key-value pair for `personDetection` and `factDetection` to specify the algorithm to be used.
Changed any red and blue text back to black.
David Vernon.
1 February 2024.

Version 2.19

Removed reference to displaying diagnostic images in Deliverable D5.2.
David Vernon.
19 June 2024.

Version 2.20

Updated Gantt charts.
David Vernon.
29 July 2024.

Version 2.21

Updated Deliverable D5.3 to reflect the different modes of operation of the `overtAttention`, including seeking functionality, and the addition of a new topic on which the mode is published. These changes are set in blue font to make them easier to see. They will be set in black in the next version.
Muhammed Danso.
29 August 2024.

Version 2.22

Adjusted wording in Deliverable D5.3 Overt Attention.
Updated D5.5.4 Robot Navigation to amend an incorrect reference to D4.2.1, replacing it with D4.2.4; to remove the key-value pair specifying the number of waypoints; and to remove the reference to using image dilation to generate a configuration space map.
The generation of the configuration space map is now be the responsibility of Task 5.5.3 Environment Map Generation. D5.5.3 has been updated accordingly.
Changed references to script language and script interpreter to robot mission language and robot mission interpreter.
David Vernon.
5 September 2024.

Version 2.23

Updated scanning mode for the `overtAttention` to prioritize detected faces when scanning the environment for general features.
Updated `gestureExecution` to include head movement to complement pointing gestures. (with the help of Adedayo)
Updated `faceDetection` to remove eye detection and add mutual gaze detection. (with the help of Yohannes)
Muhammed Danso.
20 December 2024.

Version 2.24

Completed the change of task and deliverable title from Face & Eye Detection and Localization to Face & Mutual Gaze Detection and Localization.
Updated Gantt charts.
David Vernon.
1 January 2025.

Version 2.25

Renumbered D5.5.1 to D5.5.1.1 (Gesture Executions) and added D5.5.1.2 Programming by Demonstration.
Updated Gantt charts.
David Vernon.
6 January 2025.

Version 2.26

Changed camera type form StereoCamera → RealSense for face and mutual gaze detection and localization.
Changed the configuration format from .ini → .json. for both D4.2.1 and D4.2.2
Removed incorrect statement on D4.2.1.
Removed statement regrading person detection being an optional input for the face detection.
Added statement regarding angle threshold for mutual gaze detection.
Blue text is used to mark addition of text or table entries and red text was used to mark deletion of text or table entries.
Yohannes Haile.
17 January 2025.

Version 2.27

Changed name of ROS node scriptInterpreter to behaviorController.
Use of blue and red text to flag revisions in the previous version changed to black.
David Vernon.
20 January 2025.

Version 2.28

Updated Deliverable D2.1 to include the development of a dashboard for visualizing survey responses to aid with data cleaning and forming consensus views on each element of cultural knowledge.
Updated Deliverables D4.2.1, D4.2.2, D4.2.4, and D4.2.5 to state that the simulator unit test will fail if the RealSense camera is configured.
David Vernon.
23 January 2025.

Version 2.29

Updated Deliverables D5.4.1 and D5.4.2 to include the development of C++ helper classes to read the culture and environment knowledge base files, store the knowledge, and make the knowledge accessible through a suite of access methods.
David Vernon.
25 January 2025.

Version 3.0

Revised the work plan to limit the support for the simulator. Simulator support will not be provided for any task and deliverable that involves the development of software that has the option of using an external device such as the RealSense camera or a LiDAR, or that uses the Pepper microphones, speakers, or the tablet PC, as follows.

D4.2.1 Person Detection and Localization.
D4.2.2 Face & Mutual Gaze Detection and Localization.
D4.2.3 Sound Detection and Localization.
D4.2.4 Robot Localization.
D4.2.5 Camera Calibration.
D4.3.1 Tablet PC Event.
D5.3 Attention Subsystem.
D5.5.1.2 Programming by Demonstration.
D5.5.2.1 English Text to Speech Conversion.
D5.5.2.2 isiZulu Text to Speech Conversion.
D5.5.2.3 Kinyarwanda Text to Speech Conversion.
D5.5.2.4 Integrated Text to Speech Conversion.
D5.5.3 Environment Map Generation (in SLAM mode).

Similarly, simulator support will not be provided for any task and deliverable that is dependent on data produced by one of the deliverables listed above, viz.

D4.3.2 Speech Event (dependent on D4.2.3 Sound Detection and Localization).
D5.4.3 Robot Mission Interpreter (dependent on D4.2.4 Robot Localization).
D5.5.1.1 Gesture Execution (dependent on D4.2.4 Robot Localization).
D5.5.4 Robot Navigation (dependent on D4.2.4 Robot Localization).

Consequently, the software modules that support the simulator are as follows.

D4.1 Sensor Tests.
D5.1 Actuator Tests
D5.2 Animate Behavior Subsystem.

The remaining software modules that are not impacted by the physical robot or the simulator are as follows.

D5.4.1 Cultural Knowledge Ontology & Culture Knowledge Base.
D5.4.2 Robot Mission Language.
D5.5.3 Environment Map Generation (in CAD mode).

David Vernon.
31 January 2025.

Version 3.1

Added subsections for each work package so that work packages appear in, and can be access directly from the table of contents.
Changed cultural knowledge base to culture knowledge base so that we refer consistently and correctly to the cultural knowledge kntology & culture knowledge base.
David Vernon.
10 February 2025.

Version 3.2

Clarified that languages supported in Task 4.3.2 and Deliverable 4.3.1 Speech Event are both Kinyarwanda and English, not just Kinyarwanda.
David Vernon.
25 March 2025.