Situated Dialogue for Speaking Robots

Eugénio Ribeiro
L^{2}F – Spoken Language System Laboratory
INESC-ID Lisboa / Instituto Superior Técnico
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
http://www.l2f.inesc-id.pt
eugenio.ribeiro@l2f.inesc-id.pt

ABSTRACT

This document presents a framework for the development of robots able to learn through dialogue.

The framework's architecture is a functional, layered architecture, with the mind on the top layer, competences on the middle layer and the body on the bottom layer. This architecture focuses on extensibility, flexibility and portability.

The architecture was deployed for two test scenarios, the first involving a robot able to learn different colors and identify them later and the second involving a robot able to meet people and remember their names.

Those two test scenarios were evaluated, both objectively, through previously defined evaluation tests, and subjectively, through the opinion of observers.

The evaluation had positive results and, thus, the theoretical framework was proved useful, at least in simple scenarios.

Keywords

Dialogue systems, Intelligent agents, Learning, Knowledge base, Reasoning, Robots

1. INTRODUCTION

We believe that creating a framework enabling the development of robots able to learn through dialogue may open many doors for research on Artificial Intelligence and lead to the development of robots able to help mankind in multiple tasks, from the most simple daily tasks, to difficult and dangerous ones. Furthermore, as we were not sure if the people in general had the same opinion about the utility of the framework and this kind of robots in general, we made an inquiry and found out that the great majority had the same opinion.

This led us to the development of the framework presented in this document, which had to face the following problems and accomplish the following objectives.

1.1 Problems

Enabling a robot to learn things about the surrounding environment through dialogue places many challenges, such as communicating with other agents (human or not) in the environment, identifying elements in the environment and obtaining knowledge about them, solving knowledge conflicts while reasoning about the obtained knowledge, and acting according to that knowledge.

1.2 Objectives

We defined as general objective of our work the development of a theoretical framework able to help in the development of robots able to learn through dialogue.

This framework must enable the creation of such robots, without limiting their functionality and adaptability to multiple scenarios.

It must define some high level processes for the mapping between language and knowledge, as well as for action planning adaptation according to the existing knowledge and possible actions.

Furthermore, it must provide simple adaptability to multiple and different physical or virtual bodies.

In addition to the theoretical framework, at least one test scenario must be defined and a prototype implementation of the framework for that scenario must be developed for a matter of testing and validity checking.

2. RELATED WORK

On the field of dialogue processing, there are multiple frameworks for the development of dialogue systems, like Galatea [6, 7], Olympus [2] and DIGA [9]. However, all these are based on TRIPS [1] and although there are some architectural differences, the base is ultimately the same. The TRIPS' architecture, figure 1, defines a core for the conversational system which supports asynchronous interpretation, generation, planning and acting. Also, it has a clear separation between discourse modeling and task/domain levels of reasoning. This design simplifies the incremental development of new behaviors, enhances the system's ability to handle complex domains, improves portability and allows task-level initiative. The three main components are the interpretation manager, the generation manager and the behavioral agent.

On the field of intelligent agents, in addition to the previous systems, which may also be considered agents, we found FAtiMA [5], an agent architecture with planning capabilities which uses emotions and personality to influence the agent's behavior. Its latest version is a modular architecture that separates functionalities and processes into independent components, simplifying extension. This modular architecture is composed of a core layer, figure 2, to which components can be added in order to provide additional functionality.

In the same field, we found Greta[1] [12], a conversational

[1] http://perso.telecom-paristech.fr/~pelachau/Greta/

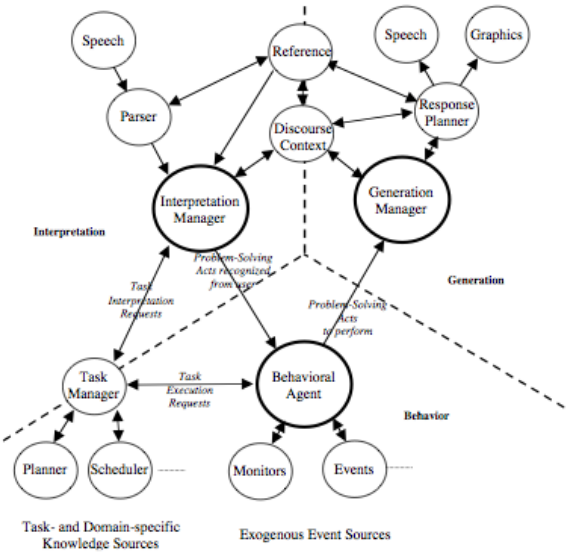

Figure 1: TRIPS Core Architecture [1]

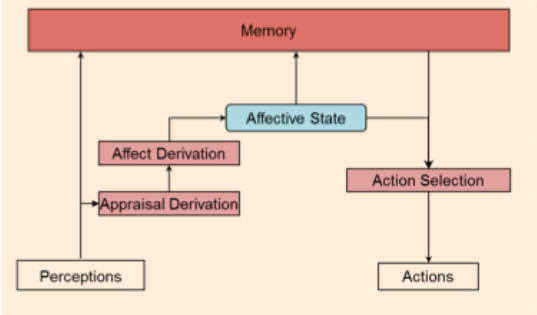

Figure 2: FAtiMA Core Architecture [5]

agent embodied in a 3D model of a woman designed to engage in information giving dialogues in the form of question / answer. Following the architecture on figure 3, when the dialogue starts, the goal in that particular domain is set and passed to the dialogue manager, enabling the creation of a discourse plan according to the agent's mind, which represents how the agent will try to achieve the communicative goal during the conversation. Once the dialogue manager selects the output, it asks the mind if there are any affective states to be activated and their intensities. In the next step, the output is enriched by the Midas module with the addition of tags indicating the synchronism between the verbal stream and other communicative functions. Finally, the enriched dialogue move is passed to the body generator, which interprets and renders it, producing the corresponding behavior.

On the knowledge representation field, we found out that ontologies are widely used for intelligent agents' knowledge representation and that there are many publicly available. Furthermore, we found two reasoners, FaCT++ [15] and Pellet [14], that are able to extract and update knowledge from those ontologies. The first focus on reasoning with classes, while the second also reasons with instances.

Finally, we also found some projects that are partially

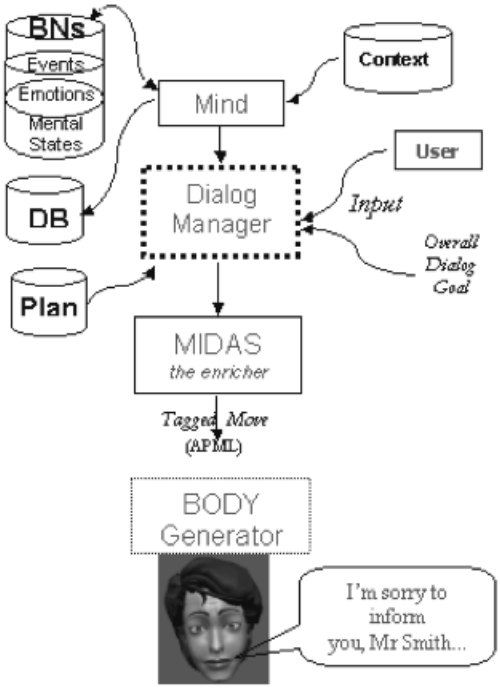

Figure 3: Greta Architecture [12]

related to ours, such as Companions[2] [11], which developed agents able to learn their owners' habits, needs and life memories; LIREC[3], which designed agents able to develop and read emotions and maintain long term relationships with humans, while acting on different platforms; CoSy[4] [4], which developed cognitive systems for multiple tasks; and CogX[5], which is the closest to ours and is devoted to developing cognitive systems able to function on open and challenging environments and deal with novelty and change.

3. ARCHITECTURE

As we are developing a framework for the development of robotic agents for multiple scenarios, it must focus on extensibility, flexibility and adaptability, as the agents may have different needs and competences, as well as multiple physical bodies, for each scenario. The LIREC[6] Project's architecture had a similar focus, thus, we took advantage of that architecture's strengths and came up with the functional, three-layer architecture presented on figure 4.

3.1 Top Layer: The Mind

On the top layer is Brainiac, the mind, which stores and reasons about knowledge, defines objectives based on that knowledge and received perceptions, and plans actions to fulfill those objectives. In order to enable Brainiac to perform these tasks we developed the Knowledge-Objectives-Competences model presented bellow.

[2] http://www.companions-project.org/
[3] http://lirec.eu/
[4] http://www.cognitivesystems.org/
[5] http://cogx.eu/
[6] http://lirec.eu/

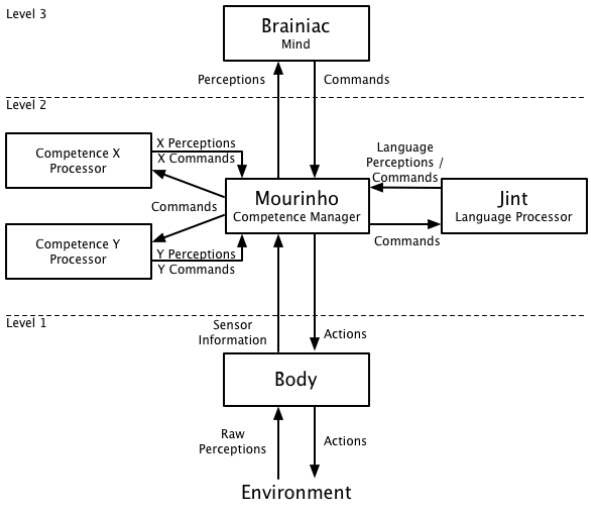

Figure 4: The system's architecture

3.1.1 The KNOC Model

The KNOC – Knowledge-Objectives-Competences – Model, is a mind model which, as the name indicates and can be seen on figure 5, has three main focuses – Knowledge, Objectives and Competences.

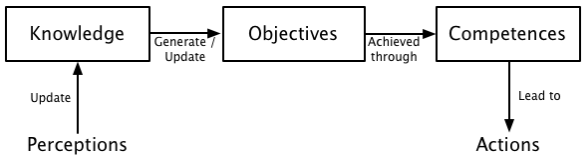

Figure 5: The KNOC Model

Knowledge management, the crucial task for a learning agent, is performed by resorting to a knowledge base which separates knowledge into two different categories – short term memory and long term memory – where the former represents the most recently received perceptions and the latter the knowledge obtained by the agent along time through experience, learning and reasoning.

As for the objectives, they are generated and updated by reasoning with both knowledge categories present in the knowledge base, a process which is able to update knowledge as a side effect.

However, reasoning is not always enough to fulfill an objective and that is where the competences become handy, as by knowing which are the competences of the current body the mind is able to generate a plan with a set of supposedly valid actions that may be executed by that body in order to fulfill an objective. Furthermore, it helps the mind deciding if an objective cannot be fulfilled with the current body and, thus, should be abandoned, at least until the body as changed.

3.2 The Manager Approach

We believe that Brainiac's functionality may be divided into four complementary high level tasks – perception processing, knowledge representation and reasoning, objective management, and action plan execution – and, thus, that an independent module should exist for each of these tasks, being responsible for its management.

As can be seen on figure 6, there are four managers – Perception Manager, Knowledge Manager, Objective Manager, and Execution Manager – responsible for each of the tasks, respectively. We believe that, together, these four managers are able to fulfill the requirements and function of the KNOC Model.

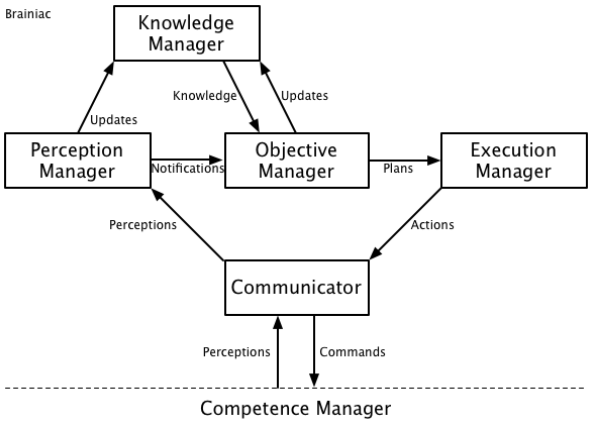

Figure 6: The Manager Approach for Brainiac

Furthermore, we also included a Communicator module that deals with communication with the competence manager, so that dependencies from specific communication interfaces and protocols are avoided in the management modules.

3.3 Middle Layer: The Competences

The middle layer aggregates the agent's competences and includes a competence manager, Mourinho, as well as multiple competence processors. As we focus on agents able to obtain knowledge through dialogue, we included a mandatory language processor, Jint, but many other processors may be added according to the scenarios and different physical bodies.

3.3.1 Mourinho: The Competence Manager

The competence manager functions as a hub with which every other component communicates, in an attempt to keep the independence between the components. It is responsible for dispatching commands from the mind and sensor information from the body to the correspondent competence processors, as well as dispatching perceptions from the later to the mind and actions to the body. Furthermore, it is responsible for knowing the current body's competences and dealing with body changes, which is achieved through a body abstraction interface, which will be presented on section 3.4.1.

As can be seen on figure 7, Mourinho is divided in three modules – Communicator, Perception Dispatcher, and Command Dispatcher – responsible for communicating with the other components, dispatching perceptions, and dispatching commands, respectively.

3.3.2 Jint: The Language Processor

Jint is a language processor, responsible for managing the speaking and listening competences of the agent. On the one hand it receives speech commands from the mind in the form

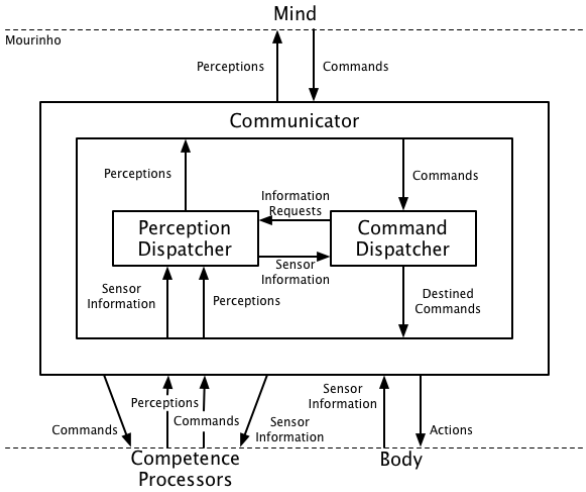

Figure 7: Mourinho: The Competence Manager

of predicates and translates them into natural language, generating the corresponding audio signal. On the other hand it receives the audio signal from the body's sensors and recognizes speech in it, generating the corresponding textual form, which is later translated into predicates and sent to the mind in the form of language perceptions.

As can be seen on figure 8, Jint has three main modules – Translator, Speech Synthesizer and Speech Recognizer. The first one is present on both the previously referred tasks, as it is responsible for translating predicates into textual natural language and vice-versa. Each of the other two modules is responsible for the other important part of the corresponding task, with the speech synthesizer being able to generate the audio signal from the corresponding textual sentence and the speech recognizer being able to do the reverse process.

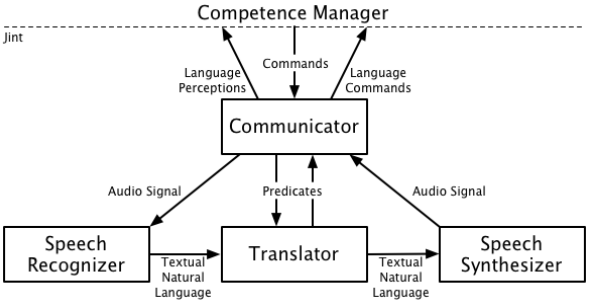

Figure 8: Jint: The Language Processor

Furthermore, Jint also has a Communicator module responsible for the abstraction of the communication interface and protocol, reducing the number of dependencies in the other modules.

3.4 Bottom Layer: The Body

The bottom layer is the agent's body, which is physical most of the times and can have multiple and distinct capabilities.

We want our framework to easily function with multiple, different bodies and even to be able to deal with body changes in runtime, However, these different bodies probably have different competences and most certainly have different interfaces. Given this, we had to come up with a way of dealing with these different bodies without compromising the system's ability to function.

The solution we found was the creation of a Body Abstraction Interface, based on the competences that a given body might have, which functions as an adapter for every body that might be used with our system.

3.4.1 Body Abstraction Interface

The Body Abstraction Interface functions, in part, as the Communicator modules presented for all the other components, as it deals with the communication with the physical body without introducing dependencies from specific communication interfaces and protocols, which is needed as they differ from body to body. However, the Body Abstraction Interface is more than that as it is able to activate only the functions that the current body is able to perform, based on competences. Furthermore, the execution of those functions is abstracted, enabling a silent transition between bodies with the same competences, even when the methodology used to achieve a certain end differs.

About competences, as they are responsible for enabling or disabling certain functions, they can help the mind decide the best way to achieve a certain objective according to the existing body. This process was previously presented on section 3.1.1.

According to the bodies we have available and the importance for our framework we selected the following set of competences and corresponding subset of functions for our version of the Body Abstraction Interface. However, new competences may easily be added, without the need for messing with instances of the interface developed for bodies where those competences are not relevant.

Moving: A competence that represents the body's ability to change its position.

- **changeSpeed(speed):** A function for changing the movement speed.
- **rotate(direction, speed):** A function to rotate in a given direction, at a given speed.
- **move(distance, direction, speed):** A function to move a given distance, in a given direction, at a given speed.

Seeing: A competence that represents the body's ability to capture visual information from the environment.

- **captureFrame():** A function that captures a single frame from the camera(s).
- **captureVideo(time):** A function that captures video input from the camera(s) during a given time period.

Hearing: A competence that represents the body's ability to capture audio information from the environment.

- **captureAudio([time]):** A function that captures audio input from the microphone(s) during a given time period or until silence is detected.

Speaking: A competence that represents the body's ability reproduce an audio signal.

playAudio(audio): A function that reproduces a given audio signal.

Displaying: A competence that represents the body's ability to display visual information.

displayFrame(frame): A function that displays a single image.

displayVideo(video): A function that displays a video clip.

displayText(text): A function that displays textual information.

It is important to notice that, although a given body has a certain competence, it is possible that it is not able to execute some of the functions from the subset enabled by that competence, leading to exception cases.s

Furthermore, it is important to refer that although our framework accepts bodies with different competences, or with no competences at all, there are some competences that are mandatory for the agent to be able to achieve its purpose of learning through dialogue. These competences, Hearing and Speaking, must exist, as they play a very important role in dialogue. Also, it is recommended that the used bodies have at least another competence that enables them to obtain information from the environment (e.g., Seeing), for the possibility of coupling linguistic information with known physical properties.

3.5 Transversal Aspects

From the previous sections we are able to conclude that each component has its independent module for communication, the Communicator module. These modules were introduced for the sake of flexibility and portability, as they remove dependencies from communication interfaces and protocols from each component's main modules.

Although any communication interface or protocol may be used, we suggest the use of ones able to perform event based, asynchronous communication as, although synchronous communication is much simpler, it leads to a severe negative impact to the system's performance.

Furthermore, as this is a framework for development, it must allow quick and simple changes to the system's configuration, in order to simply testing and the comparison of multiple solutions for a given problem. Also, the framework must provide a simple way of defining which actions and perceptions might occur in a given scenario, so that transition between scenarios is simplified.

In order to achieve this, we opted for the usage of configuration files, as they can be easily edited by the users of the framework. Also, we decided that each component must have a default configuration and that other configurations might be loaded by the users, from different locations, so that there is always a backup configuration present.

4. SCENARIOS

In order to test the value and validity of the theoretical framework presented on the previous chapter, we defined two test scenarios involving learning through dialogue. The first scenario is about a robot able to learn colors through dialogue with a human teacher, while the second is about a robot that meets people and is able to remember their names. Although these are relatively simple scenarios, with small, closed domains, they are both focused on learning through dialogue and, thus, are valid test cases for the framework's ability to deal with such kind of learning. Furthermore, they have different learning focus and domains and, thus, are also valid test cases for the framework's extensibility, flexibility and adaptability.

4.1 Scenario 1: Learning the Colors

In this scenario, a robotic agent is taught different colors by a human teacher and should, after that, be able to identify those colors when prompted. Furthermore, after learning a given color, the agent should be able to select the object (ball in our example) of the correct color when prompted for the one of the given color, amongst multiple choice possibilities.

A possible interaction for this scenario is the following:

USER: What color is this?
[*User shows a yellow paper sheet to the robot.*]
PHASTER: Sorry, I don't know. What color is it?
USER: It's yellow.
PHASTER: Ok, now I know. Thanks!
USER: What about this? What color is it?
[*User shows a blue paper sheet to the robot.*]
PHASTER: I don't know that color. What color is it?
USER: It's blue.
PHASTER: Thanks, now I know one more color.
USER: Can you guess this color now?
[*User shows an orange paper sheet to the robot.*]
PHASTER: I believe it's yellow. Am I correct?
USER: You were close, but no.
PHASTER: Oh, what color is it then?
USER: Orange.
PHASTER: Ok, I'll remember that.
USER: Now, finally, tell me what color this is.
[*User shows a blue paper sheet to the robot.*]
PHASTER: Oh, I'm sure that's blue!
USER: Great!
[*User places three balls of different colors in front of the robot.*]
USER: Find the yellow ball!
[*The robot rotates until it is facing the yellow ball.*]
PHASTER: Here it is!
USER: Nice!

4.2 Scenario 2: Acquaintances

In this scenario, a robotic agent meets multiple people and learns their names through conversation. Later on, if a known person greets the agent, it should be able to greet that person, using the correct name.

A possible interaction for this scenario is the following:

PERSON 1: Hi!
PHASTER: Hi! I'm Phaster. What's your name?
PERSON 1: Eugénio.
PHASTER: Nice to meet you, Eugénio!
PERSON 2: Hello Phaster!
PHASTER: Hi! Who are you?
PERSON 2: I'm David.
PHASTER: Hi David!
PERSON 1: Hi again!
PHASTER: Hi Eugénio!

4.3 Competences

By analyzing the two scenarios, we decided that, in order to perform as expected, the agents should have hearing, speaking, seeing and moving competences. Given this, we defined the two following predicate sets, the first defining a set of high-level actions the agents may perform and the second defining a set of high-level perceptions the agents may capture.

4.3.1 Actions

Rotate(direction, speed): Rotate the body at the given speed until it is facing the given direction.

Move(distance, direction, speed): Moves the body a given distance, in the given direction, at the given speed.

Find(object, properties): Tries to find the properties of the given object.

State(object, properties, certainty): States the properties of the given object, with a relative certainty.

Ask(object, properties): Asks for properties of a given object.

Greet([person]): Greets someone, using the person's name if it is given.

Thank([person]): Thanks someone, using the person's name if it is given.

4.3.2 Perceptions

Crash(): States that the robot has collided with something.

Color(red, green, blue): A color represented as its red, green and blue components.

Circle(x, y, radius, color): A circle represented has its center's x and y coordinates, as well as its radius and color in RGB.

Face(id): A face represented by an id calculated by the face recognizer. The reasons for this kind of representation will be explained later.

Statement(object, properties): A statement about the properties of an object.

Question(object, properties): A question about the properties of an object.

Request(object, properties): A request for an object with the given properties.

Greeting(): A greeting.

Negation(): A negation(e.g. no) negating something.

Affirmation(): An affirmation(e.g. yes) confirming something.

5. DEPLOYMENT

With the scenarios defined, we developed an implementation of the framework for testing its performance in the two scenarios. As this was simply a prototype, designed for testing purposes on specific scenarios, it is important to notice that some of the framework features, which were included for further extensibility, flexibility and adaptability were neglected in this implementation for simplicity purposes.

5.1 Brainiac

While designing our Brainiac implementation, we though about using OWL ontologies for knowledge representation and Pellet [14] as a reasoner and, thus, we decided to develop the managers in the Java programming language for a matter of compatibility. However, later, during development, we noticed that the kind of knowledge stored and reasoned with in our scenarios was too simple for the need of such an evolved reasoner and, thus, we decided to represent and reason with it in simpler ways, which will be presented later in this section.

5.1.1 Perception Manager

For the Perception Manager, we included a sub-module, the Perception Storage, which stores the asynchronously received perceptions until a new iteration of the KNOC cycle starts. When a new iteration starts, the main module gets those perceptions and processes them, selecting the important ones and discarding the others, using a set of restrictions.

After this process, the important perceptions are sent to the Knowledge Manager as updates to short-term memory and notifications are sent to the Objective Manager informing it of the existence of new knowledge which may be useful.

5.1.2 Knowledge Manager

As short-term memory stores only the most recent important perceptions, we opted for storing the perceptions themselves, which are replaced every time a new perception of the same type is received.

For the representation of associations between color codes and respective names, we defined a structure called *Color Cluster*. Each of these clusters is associated with a different color name and has a representative color code and a maximum deviation.

Finally, we wanted our agents to be able to associate a name with any person they met. However, we were not able to find a face recognition system able to perform online training and, as this was not part of our focus, we opted for using a pre-trained face database, together with the Eigenfaces [16] algorithm for face recognition. This approach reduced the agents' learning process to a simple association between the identifier given by the face recognizer and the name obtained through dialogue. Given this, for the representation of the knowledge obtained through the learning process, we opted for a simple hash map, mapping face recognizer identifiers into names obtained through dialogue.

5.1.3 Objective Manager

The Objective Manager generates and updates objectives according to the existing knowledge and generates action when that knowledge is not enough. For our two scenarios we defined three simple kinds of objective - Answer, Find, and Greet.

Although there are only three, limited and specific kinds of objective, they are enough to achieve the expected functionality for our scenarios. This is one example of the previously referred neglections of extensibility, flexibility and adaptability for the sake of simplicity, as this implementation is just a simple testing prototype.

These objectives are generated and updated according to the received perceptions and were defined so that, in friendly conditions, only one action at most is needed for obtaining

enough knowledge to change each objective's state.

5.1.4 Execution Manager

As the transition between states of the previously defined objectives requires the maximum of one action, we decided not to store the action plans and execute the actions immediately instead. This way, the Execution Manager receives an action to be executed from the Objective Manager, verifies if there is a similar one in the action history and, in the negative case, stores it in the history and dispatches it to the competence manager as a command.

5.2 Mourinho

In contrast with the complexity of the mind, the competence manager is a simple dispatcher, able to select which commands belong to each processor and to send the perceptions they capture to the mind. Given the specificity of the scenarios, this is a simple task, with the commands being dispatched as follows:

- Ask / State / Greet / Thank → Language Processor
- Find → Vision Processor
- Rotate / Move → Mapped into actions → Body

Also, input from the body sensors is redirected to the competence processors and, on the other way around, actions sent by the processors are redirected to the body.

As Mourinho is constantly processing commands and perceptions, it was developed in C++ for the sake of performance. Furthermore, the whole process is event based, with multiple threads being created for dispatching the received messages when they arrive.

5.3 Jint

For the language processor, we decided to use English as the agent's language, thus, we had to find both an English recognizer and an English synthesizer. We decided to use CMU Sphinx[7] [8] for recognition and MARY TTS[8] [13] for synthesis, as they are widely used and have acceptable performance and results. As they both are easier to use with the Java programming language, we developed the Translator module for the same language.

For the Speech Recognizer module we developed a simple grammar, as explained on the CMU Sphinx website, defining the sentences we were expecting to capture in our scenarios. With this grammar loaded, the CMU Sphinx speech recognizer receives the audio input which was directed from the body to Jint by the competence manager and is able to capture the sentences matching the grammar, generating the corresponding textual representation.

For the Speech Synthesizer, it is necessary to configure MARY TTS by selecting the language and kind of voice to be used. After that it is enough to send the textual sentences received from the Translator module to MARY TTS for it to generate the corresponding audio signal.

For the Translator module we had to develop both an English Interpreter and an English Generator, mapping the textual sentences into predicates and vice-versa, using a bag of words approach for the first task and simple generation rules for the second.

[7] `http://cmusphinx.sourceforge.net/`
[8] `http://mary.dfki.de/`

5.4 vPro

As the learning focuses of our scenarios are colors and people's names, we decided that the best way to capture both color and face information that could be associated with linguistic information was through vision, that is, by capturing visual features in the video captured by a camera. To achieve this, we added a new competence processor to our architecture for the vision competence. This vision processor, vPro, as can be seen on figure 9, consists on an agglomerate of feature detectors which receive the video frames and detect possible elements of the corresponding feature in those frames.

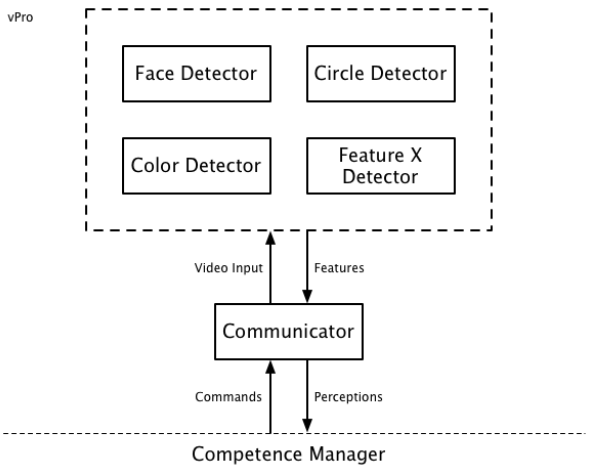

Figure 9: vPro: The Vision Processor

For our scenarios we decided to include detectors for colors, circles and faces, however, vPro is prepared for the simple addition of new ones. These detectors analyze the video frames and capture the corresponding features if present.

Similarly to the competence manager, the vision processor also has performance issues that must be thought of, as it must quickly process video frames to find features. So, we also opted for the C++ programming language to implement it. Furthermore, the OpenCV[9] [3] library already provides multiple feature detectors that can be used as is, or combined to produce more evolved detectors, and has a C++ version, thus, we decided to take advantage of its capabilities while implementing vPro.

5.5 Body

As a body, we decided to use Magabot[10], a platform designed to give mobility to laptops, as together with a laptop it provides us the sensors we need – camera, microphone, and crash bumpers – as well as the actuators – speakers and wheels. Furthermore, it has the capability to support other structures on top of it, so that height can be adjusted according to the scenario. The physical appearance of our agent can be seen on figure 10.

Using this body's sensors and actuators, together with both competence processors, we were able to implement the Body Abstraction Interface with the previously presented required competences.

[9] `http://opencv.willowgarage.com/`
[10] `http://magabot.cc`

Figure 10: Phaster, our learning robot

5.6 Communication

Due to the use of different programming languages in the different system components, we decided to use the YARP[11] [10] framework for implementing the Communicator modules, as it provides a simple, asynchronous, inter-language communication interface through the use of its *Port* and *Bottle* objects.

5.7 Configuration

Finally, for configuration, we decided to take advantage of the Java programming language Properties for the components written in that language, while for the components written in C++ we developed our own configuration file reader. In both cases, the configurations are written in the form *object.property = value*.

6. EVALUATION

In order to evaluate each scenario, we performed some objective tests that evaluate the agent's performance in that scenario, as well as subjective tests where observers give their opinion about the agent's learning skills.

There were 65 observers, with ages ranging between 17 and 40 years, with most of them assuming to have a close relationship with technology.

After observing each scenario the observers answered two questions:

1. Do you think that the robot was able to learn?
2. Do you think that the dialogue was natural?

6.1 Scenario 1: Learning the Colors

For the first scenario we used the following evaluation procedure:

1. Teach five different colors to the robot using sheets of paper.
2. Count the number of turns until the agent can guess them all correctly.
3. Ask the color of five different objects with known colors.
4. Count the number of correct answers.
5. Place three balls of different colors in front of the robot.
6. Ask the agent to select the ball of a given color.
7. Verify if the agent selected the correct ball.

By repeating this procedure ten times, we obtained the results presented on figure 11. As we can see, only one time more than two learning turns were needed for the agent to be able to correctly guess all the colors, which we believe are good results. Furthermore, when the agents was prompted about the color of different objects, the average number of correct guesses was 4.2 out of 5, never failing more than two guesses, which we consider to be very good results. As for the results of the ball selection test, they are not presented as they were inconclusive, due to problems with the circle detector usage, which led to no balls being detected on most of the tests. However, on the few relevant tests, the agent was able to select the right ball.

	Learning Turns	Correct Guesses
Round 1	1	4
Round 2	2	5
Round 3	2	4
Round 4	2	5
Round 5	1	3
Round 6	1	4
Round 7	3	5
Round 8	2	3
Round 9	1	4
Round 10	2	5
Average	1.7	4.2
Maximum	3	5
Minimum	1	3

Figure 11: Scenario 1: Test results

An early stage prototype for this scenario was publicly presented on a robotics board part of the Game On[12] exposition. Although this was a very simple prototype and it still did not include speech recognition, the public reacted very well to it, understanding what we were trying to show and proposing multiple applications outside the color domain.

Furthermore, the public stated that if robots were able to learn through dialogue with open domain, the number of common people wanting to intellectually develop their own robots would highly increase, as there would be no need for knowledge of complicated programming languages. However, it was also stated that speech recognition was a must for this kind of robot, as it is much simpler for humans to speak than to write, leading to a much more pleasant experience for the human users.

As for the observer's opinion, 92% of the observers believe that the robot was able to learn. However, the other 8% believe the contrary, with reasons ranging from believing that the agent was just randomly guessing, to the difficulty of defining what might be considered learning.

[11] `http://eris.liralab.it/yarp/`

[12] `http://gameon.gameover.sapo.pt/`

As for the dialogue being natural, 88% of the observers agreed, while the other 12% disagreed, stating that the vocabulary was limited and that it will always be like that, as it must be programmed.

6.2 Scenario 2: Acquaintances

For this scenario we expected to use an evaluation procedure involving at least two people, however, due to the previously referred limitations of the face recognizer and lack of availability from possible users, we had to use the following single person procedure:

1. The person greets the agent.
2. When prompted, the person introduces himself to the agent.
3. The person leaves the room.
4. The person comes back and greets the agent.
5. Verify that the robot remembers the person's name.

This procedure was repeated ten times and in all of them the agent was able to notice the presence of a person and remember the correct name.

For the question about the robot's learning skills, the results were slightly lower than the ones from the previous scenario, with only 89% believing that it was able to learn. The lower value is due to the fact that there was only one person interacting with the agent and, thus, the observers were unable to understand if it could distinguish between two different people.

However, for the dialogue question, the results were slightly higher than on the previous scenario, with 89% of the observers stating that the dialogue was natural. As reasons for not agreeing, the only one different from the ones stated in the previous scenario was that the agent did not show emotional affection towards the person.

7. CONCLUSIONS

From the previous sections we can conclude that agents developed using our framework are able to learn through dialogue, at least in relatively simple scenarios.

As it was developed with focus on extensibility, flexibility and adaptability it is simple to adapt the framework to multiple scenarios, through the addition or extension of actions, perceptions and competence processors.

Furthermore, in our test scenarios, the performance was good, with fast action and response times. Also, the accuracy in the evaluation tests was really high.

As for the opinion of the observers, it was positive for the large majority. However, there were also some negative opinions, which can help us improving our system in the future.

Continuing on the negative side, some of the vision processor's detectors did not work as expected. Furthermore, speech interpretation and generation, as well as knowledge management, were not very developed and scenario specific, which, in some ways, limited the developed agents. However, we believe that this is normal, as for the development of such a large framework in such a short time, some parts of its deployment must be neglected.

In our opinion, we were able to fulfill the expectations, at least partially and we believe that the previously presented results show that the great majority of the observers is of the same opinion.

8. FUTURE WORK

In the most immediate future, we want to improve the vision processor, by correcting the circle detector and developing a face recognizer able to perform online training.

Also, in the near future we want to improve speech interpretation and generation, as well as knowledge management, in order to allow more open scenarios and dialogues.

Furthermore, we want to test the framework in more complex scenarios to be aware of its true value. One of the scenarios we are more eager to develop is one where the agent is able to learn information about itself and use that information to perform new actions.

On a different area, we would also like to add emotional components to our system, as we believe it can make the dialogue more natural and the agents more appealing to the humans interacting with them.

Acknowledgements

First of all, we would like to thank everyone who spent some precious time answering our inquiries. Furthermore, we would like to thank Filipa Menezes for giving us the opportunity to present our prototype at the Game On exposition and Artica[13] for providing us the Magabot we used to develop our prototype.

9. REFERENCES

[1] James Allen, George Ferguson, and Amanda Stent. An Architecture For More Realistic Conversational Systems. In *6th international conference on Intelligent user interfaces*, pages 1–8, 2001.

[2] Dan Bohus, Antoine Raux, Thomas K. Harris, Maxine Eskenazi, and Alexander I. Rudnicky. Olympus: an open-source framework for conversational spoken language interface research. In *HLT-NAACL 2007 workshop on Bridging the Gap: Academic and Industrial Research in Dialog Technology*, 2007.

[3] Gary Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[4] Henrik Iskov Christensen, Geert-Jan M. Kruijff, Aaron Sloman, and Jeremy L. Wyatt. *Cognitive Systems*. Springer, 1st edition, 2010.

[5] Joao Dias, Samuel Mascarenhas, and Ana Paiva. FAtiMA Modular: Towards an Agent Architecture with a Generic Appraisal Framework. 2011.

[6] Shinichi Kawamoto, Hiroshi Shimodaira, Tsuneo Nitta, Takuya Nishimoto, Satoshi Nakamura, Katsunobu Itou, Shigeo Morishima, Tatsuo Yotsukura, Atsuhiko Kai, Akinobu Lee, Yoichi Yamashita, Takao Kobayashi, Keiichi Tokuda, Keikichi Hirose, Nobuaki Minematsu, Atsushi Yamada, Yasuharu Den, Takehito Utsuro, and Shigeki Sagayama. Open-source Software For Developing Anthropomorphic Spoken Dialog Agents. In *PRICAI-02, International Workshop on Lifelike Animated Agents*, pages 64–69, 2002.

[7] Shinichi Kawamoto, Hiroshi Shimodaira, Tsuneo Nitta, Takuya Nishimoto, Satoshi Nakamura,

[13] `http://artica.cc`

Katsunobu Itou, Shigeo Morishima, Tatsuo Yotsukura, Atsuhiko Kai, Akinobu Lee, Yoichi Yamashita, Takao Kobayashi, Keiichi Tokuda, Keikichi Hirose, Nobuaki Minematsu, Atsushi Yamada, Yasuharu Den, Takehito Utsuro, and Shigeki Sagayama. Galatea: Open-source Software for Developing Anthropomorphic Spoken Dialog Agents. In *Life-Like Characters: Tools, Affective Functions, and Applications*, Cognitive Technologies, pages 187–212. Springer-Verlag, 2004.

[8] Kai-Fu Lee, Hsiao-Wuen Hon, and Raj Reddy. An Overview of the SPHINX Speech Recognition System. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 38(1):35–45, 1990.

[9] Filipe Miguel Fonseca Martins. DIGA - Desenvolvimento de uma plataforma para criação de sistemas de diálogo. Master's thesis, Instituto Superior Técnico, September 2008.

[10] Giorgio Metta, Paul Fitzpatrick, and Lorenzo Natale. YARP: Yet Another Robot Platform. *International Journal on Advanced Robotics Systems*, 3(1):43–48, 2006.

[11] Oli Mival and David Benyon. Introducing the Companions Project: Intelligent, Persistent, Personalised Interfaces to the Internet. In *The 21st British HCI Group Annual Conference (HCI 07)*, 2007.

[12] Catherine Pelachaud, Valeria Carofiglio, Berardina De Carolis, Fiorella de Rosis, and Isabella Poggi. Embodied Contextual Agent in Information Delivering Application. In *First International Joint Conference on Autonomous Agents and Multi-Agent Systems*, pages 758–765, 2002.

[13] Marc Schröder and Jürgen Trouvain. The German Text-to-Speech Synthesis System MARY: A Tool for Research, Development and Teaching. *International Journal of Speech Technology*, 6:365–377, 2003.

[14] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A Practical OWL-DL Reasoner. *Web Semantics*, 5(2):51–53, 2007.

[15] Dmitry Tsarkov and Ian Horrocks. FaCT++ Description Logic Reasoner: System Description. In *International Joint Conference on Automated Reasoning (IJCAR 2006)*, pages 292–297, 2006.

[16] Matthew Turk and Alex Pentland. Eigenfaces for Recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, 1991.