Spoken Language Processing in a Conversational System for Child-Robot Interaction

Ivana Kruijff-Korbayová[1], *Heriberto Cuayáhuitl*[1], *Bernd Kiefer*[1], *Marc Schröder*[1]
Piero Cosi[2], *Giulio Paci*[2], *Giacomo Sommavilla*[2], *Fabio Tesser*[2]
Hichem Sahli[3], *Georgios Athanasopoulos*[3], *Weiyi Wang*[3], *Valentin Enescu*[3], *Werner Verhelst*[3]

[1]DFKI GmbH, Language Technology Lab, Saarbrücken, Germany
[2]Istituto di Scienze e Tecnologie della Cognizione, ISTC, C.N.R., Italy
[3]Interdisciplinary Institute for Broadband Technology - IBBT,
Vrije Universiteit Brussel, Dept. of Electronics and Informatics, Belgium
{ivana.kruijff, heriberto.cuayahuitl, kiefer, marc.schroeder}@dfki.de
{piero.cosi, giulio.paci, giacomo.sommavilla, fabio.tesser}@pd.istc.cnr.it
{hsahli, gathanas, wwang, venescu, wverhels}@etro.vub.ac.be

Abstract

We describe a conversational system for child-robot interaction built with an event-based integration approach using the Nao robot platform with the Urbi middleware within the ALIZ-E project. Our integrated system includes components for the recognition, interpretation and generation of speech and gestures, dialogue management and user modeling. We describe our approach to processing spoken input and output and highlight some practical implementation issues. We also present preliminary results from experiments where young Italian users interacted with the system.

Index Terms: human-robot interaction, integration, Nao, Urbi, italian children speech recognition, italian speech synthesis, voice activity detection, sound source localization, dialogue management, natural language generation, non-verbal behavior generation.

1. Introduction

Children are keen users of new technologies. And new technologies can provide interesting opportunities to enrich children's experience, for example for educational and therapeutic purposes [1]. Since children are not small adults, it is necessary to research their specific needs and develop systems that address these needs [2], [3]. To this end we are building a conversational system for child-robot interaction in the ALIZ-E project [4], a multi-partner consortium focussed on the design of long-term, adaptive social interaction between robots and child users in real-world settings. The goal of the ALIZ-E project is to develop embodied cognitive robots for believable any-depth affective interaction with young users over an extended and possibly discontinuous period.

The development of a conversational human-robot interaction system involves the integration of potentially many components and ensuring proper interaction and synchronization between them. While most work in spoken dialogue system development is based on pipeline architectures, there are notable exceptions such as [5, 6], which execute system components in parallel (weakly-coupled or tightly-coupled architectures). We presented our event-based approach to system integration in [7] and described the components constituting the first version of our system. The current paper is based on the next version. We highlight some practical issues concerning spoken language processing that arise from the context and purpose of use of the robot, and motivate the employed technologies.

Our system uses the Nao robot (Fig.1) and we implemented three game-like activities that a child can undertake with it:

- quiz: the child and the robot ask each other series of multiple-choice quiz questions from various domains, and provide evaluation feedback;
- imitation: either the child or the robot presents a sequence of simple arm poses that the other tries to memorize and imitate;
- dance: the robot teaches the child a sequence of dance moves

Figure 1: The Nao robot in the measurement setup in a sound lab at VUB (top left) and playing the Quiz game (top right), Imitation game (bottom left), and Dance game (bottom right).

In Section 2 we give an overview of the integrated system. In Sections 3–6 we focus on the processing of spoken input and output. We describe our approach and highlight some practical implementation issues. We point out changes from the previous version of the system and explain the reasons that lead to them. In Section 7 we explain our event-based approach to system integration on concrete examples. In Section 8 we present preliminary evaluation results using data collected in experiments

with Italian children. In Section 9 we conclude and present an outlook.

2. The Integrated System

Fig. 2 depicts the components of the system. For their integration we use the Urbi middleware [8] (cf. Section 7 for details).

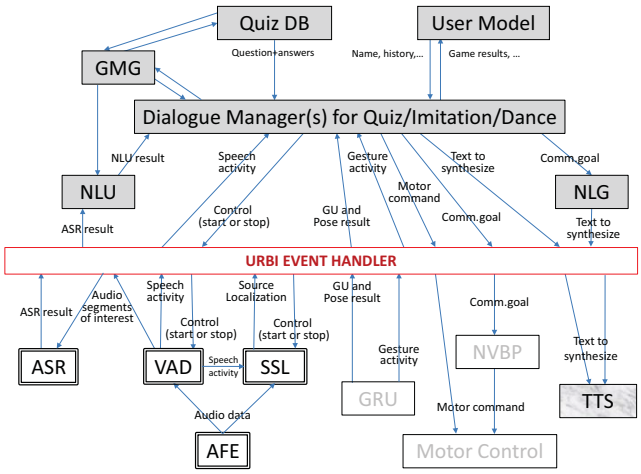

Figure 2: The components of the integrated system. Filled boxes indicate components implemented in Java, double-line boxes C/C++, and plain boxes UrbiScript. The TTS component with the marble-filled box is either the Acapela TTS on the Nao or the Mary TTS implemented in Java.

The Voice Activity Detection (VAD), Sound Source Localization (SSL) and Audio Front-End (AFE) components perform speech signal detection and capture (Section 3). The speech input is then processed by Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) (Section 4). Visual input is processed by the Gesture recognition and Understanding component (GRU). For example, to recognize the poses in the imitation game it uses face detection and either motion detection or various optical flow algorithms. Speech and gesture interpretations go to the Dialogue Manager (DM) that bears primary responsibility for controlling the robot's conversational behaviour as well as the progress of the game at hand (Section 5). It maintains a system state representation and selects the next system action. How exactly this is done depends on the game. For example, selecting the next suitable question in the quiz game is done by a separate Game Move Generator (GMG) component that also accesses the Quiz Database of questions and answers. User-specific information (e.g., name, age) and interaction history (e.g., games played, achieved performance) are kept in the User Model, which also receives updates from the DM. Spoken output is produced by the Natural Language Generation (NLG) and Text-To-Speech Synthesis (TTS) components (Section 6). The Non-verbal Behavior Planning (NVBP) and Motor Control (MC) components produce arm gestures and head&body poses. Besides the game-specific moves and poses in the imitation and dance games, static key poses are produced to display emotions, namely anger, sadness, fear, happiness, excitement and pride [9].

The following sections present in more detail the components for spoken language input and output processing.

3. Speech Signal Detection and Capture

Ultimately the children and the robot should move around freely, in a standard environment with unpredictable background noise. No particular instrumentation, such as a head-mounted michrophone, should be required. The following components are developed towards this goal.

3.1. Audio Front End (AFE)

The AFE component captures the speech signal from the microphones, makes preliminary preprocessing such as sample rate conversion, and sends the audio buffers to the VAD component.

3.2. Voice Activity Detection (VAD)

The VAD module is important to facilitate the spoken interaction. It allows the robot to detect that dynamically varying sound sources which could be of interest for further analysis (such as e.g. human speech) are active. Our studies indicate that most VAD algorithms described in the literature are greatly affected by the type of background noise. Motivated by the fact that the Nao robot will interact within an environment of unknown background noise conditions, we have examined the performance of different VAD algorithms for different background noise types that exhibit certain characteristics and proposed a robust energy based VAD algorithm. This VAD algorithm uses the feature of smoothed energy contained in the frequency region of interest and can be configured for different background noise types. We showed that it outperforms conventional spectrum based VADs under certain noise conditions and for certain configurations [10]. The VAD output is communicated to other components via the Urbi event emission functionality.

3.3. Sound Source Localization (SSL)

The ability of the robot to localize the direction where a sound is coming from is important for context-awareness. Robot audition systems often utilize microphone arrays consisting of many microphones to increase the SSL performance. The time delay on arrival (TDOA) is usually estimated under the assumption that the microphone array is located in free space. These conditions are not fulfilled in real-world robots such as Nao, where a small number of microphones are mounted on the robot's head cover. Scattering of the sound wave along the shape of the robot has a significant influence on the conventional TDOA estimation. In order to address this, we implemented a Generalized Cross-Correlation (GCC) based SSL method which utilizes a set of pre-measured TDOAs, followed by parabolic interpolation [11]. Moreover, our experiments highlight the importance of speech denoising for performing SSL under highly noisy conditions. The effect of different denoising techniques as GCC weighting functions has been studied and their performance has been assessed under internal and ambient noise types. Finally, we have demonstrated that taking Nao's microphone frequency responses into account in the GCC weighting function can considerably improve the SSL accuracy [12].

4. Spoken Input Processing

The acoustic and linguistic characteristics of child speech differ widely from those of adult speech, not only at the frequency range level but also in the production modalities [13]. For example, there are more dysfluences not only in spontaneous speech but even in read speech. [14] showed that word error rate for children is normally higher than that for adults even when using

an acoustic model trained on child speech. Developing components for robust spoken input processing for children is thus of crucial importance in ALIZ-E].

4.1. Automatic Speech Recognition (ASR)

The first version of our system used CMU Sphinx-3 [15]. It has however proven difficult to implement live decoding and run-time features with it and the Sphinx-3 upstream code is no longer maintained. We thus switched to the Open-Source Large Vocabulary CSR Engine Julius [16] as the ASR engine.

4.1.1. Sphinx/Julius comparison

Table 1 provides a comparison of Sphinx-3 and Julius4. The following points in particular have driven our decision to change the ASR engine:

- Julius decoder API is very well designed (it made integration smoother in comparison with Sphinx-3).
- Its system requirements are low (this is important in an integrated system handling several components).
- Language models can be swapped at run-time.
- Configuration is modular.
- Multi-model recognition is possible.
- On-the-fly Input/AM normalisation is available.

Feature	Sphinx-3	Julius4
Open Source	yes	yes
System requirements	Computation and memory intensive (Each word has its own HMM)	Low memory requirement: <32MB for work area (<64MB for 20k-word dictation with on-memory 3-gram LM)
Decoder API	no	yes
Decoder binary	yes	yes
AM formats	Sphinx	HTK
AM training	Sphinx	HTK
LM formats	ARPA N-gram Finite State Grammar	ARPA N-gram DFA grammar isolated word user-defined functions
Configuration	Monolithic Held for the entire execution	Modular Run-time swapping allowed
Parallel Multi-model recognition	no	yes
Confidence scoring	yes	yes
Integrated VAD	yes	yes
Input/AM normalisation	MLLR VTLN MAP	MLLR (ext. tools) VTLN CMN CVN
Input/AM normalisation (on-the-fly)		VTLN CNN CVN
Output	N-best Word lattice	N-best Word lattice Confusion network

Table 1: Sphinx-3/Julius4 features comparison

With Julius, it has indeed proven very easy to implement the desired features and to integrate them into the system.

4.1.2. Training of Acoustic Models for Children Speech

The Julius distribution does not include specfic training tools for acoustic models, however any tool that creates acoustic models in the Hidden Markov Model Toolkit (HTK) format can be used. We used the HTK tools [17], following the Voxforge HTK training for Julius tutorial [18]. Using the same procedure we trained an Italian adult acoustic model, based on the training data provided for the EVALITA 2011 Forced Alignment task [19] (a subset of the CLIPS corpus) and an Italian child acoustic model, using the FBK CHILD-IT corpus [20].

Since the system needs to carry out interaction with children in Italian, we built a corresponding Julius acoustic model using ChildIt [20], an Italian children voice corpus that contains almost 10 hours of speech from 171 children, following the same strategy we have previously adopted in several tests training and decoding ChildIt with Sphinx and other ASR tools [21]. We plan to strengthen ASR accuracy by recording an audio plus transcription corpus, thus resulting in an improved child voice acoustic model. Also, we plan to apply Speaker Adaptation techniques, using a small amount of observed data from individual speakers to improve our speaker-independent model. Adaptation can be very useful in the ALIZ-E project because children's vocal tract lengths can differ a lot. Since each child will interact several times with the robot we can consider using data from previous interactions for adapting the models.

4.1.3. LM training

Julius supports N-gram, grammar and isolated word Language Models (LMs). Also user-defined functions can be implemented for recognition. However its distribution does not include any tool to create LMs, with the exception of some scripts to convert a grammar written in a simple language into the Deterministic Finite Automaton (DFA) format needed by the engine. External tools thus need to be used to create an LM.

Since Julius supports N-gram LMs in ARPA format, we used the SRI-LM toolkit [22] to train a simple model for recognition of questions in the quiz game, using the Quiz Database as training material. The model is very simple and limited, but it should suffice to recognise properly read questions (they are expected to be from the training set), especially if used in conjunction with some other, more flexible, model.

The Julius engine distribution also includes tools that allow to express a grammar in a simple format and converting it to the Julius DFA format. That format, however, has very few constructs for writing a proper grammar by hand and writing a non-trivial grammar is very hard. Third-party tools exist to convert an HTK standard lattice format (SLF) to the DFA format and to optimise the resulting DFA [16]. SLF is not suitable to write a grammar by hand, but HTK provides tools that allow a more convenient representation based on the extended Backus-Naur Form (EBNF) [17]. We wrote a simple grammar for quiz answer recognition in the EBNF-based HTK grammar language. Part of the grammar was automatically derived by including the answers from the Quiz Database. Several rules were added to handle common answers and filler words.

4.2. Natural Language Understanding (NLU)

In order to achieve robustness against ASR errors and incomplete or ungrammatical sentences, speech input interpretation in our system proceeds along two paths: For the recognition of quiz questions, answer options and answers we use fuzzy matching of content words contained in the ASR hypothesis against the entries in the quiz database. Any other speech input is interpreted using our partial parsing approach.

The partial parsing approach works as follows. The N-best list from ASR is recombined into a compacted lattice before parsing proper to avoid re-analysing common subsequences. NLU does not use the word graph from the main decoder di-

rectly because it does not contain the LM scores which are helpful to guide parsing. Several heuristics are applied to merge similar edges or nodes to reduce lattice size and thereby the parsing effort. The effects of these heuristics on parsing performance and accuracy still have to be measured with real data. The compacted lattice is then directly analysed by an agenda-based chart parser, which uses a hand-written competence grammar based on the Multimodal Combinatory Categorial Grammar framework [23] implemented in OpenCCG [24]. The agenda, together with an appropriate search strategy, allows to focus parsing on the promising parts of the input, and thus aims to find the best partial analyses. Currently, the scores from the speech recognizer are used to guide the search. It is planned to enhance this by using a statistical model based on the CCG grammar and on information coming from the dialogue manager to include as well grammatical as real-world information.

5. Dialogue Manager (DM)

The task of the DM is to keep track of the state in which the interaction is, to integrate the interpretations of the user's spoken input (or nonverbal actions) w.r.t. this state, and given this state select the next (communicative) action of the system as a transition to another state, making progress towards a goal. The first version of our system used a finite state dialogue model. However, our pilot experiments showed that children tend to behave unpredictably [25], and the system therefore needs to be very flexible. Moreover, different children have very different interaction styles (e.g., some are shy while other take a lot of initiative), and the system should be adaptive to this.

Recent years have seen a boom in research in spoken dialogue management using probabilistic methods [26, 27, 28, 29] and optimisation of dialogue policies using reinforcement learning [30]. We experiment with these methods in the current version of the system for the quiz and imitation game. Our DM applies the learning approach developed in [31] and [32], which extends the state representation of Markov Decision Processes with relational representations to enable compact representations of large state spaces, and belief states to handle uncertainty. It uses a hierarchy of reinforcement learning agents that learn their dialogue policies from a simulated environment (partially estimated from data). Such policies specify a mapping from dialogue states describing situations in the interaction, to (communicative) actions roughly corresponding to dialogue acts.

In order to allow for flexibility in cases where a user takes initiative and strays from the interaction flow expected by the policy, we developed an approach for flexible hierarchical dialogue control which relaxes the top-down execution of subdialogues, by supporting combined hierarchical and graph-based execution [33]. The interactions become less rigid because it follows a partially specified hierarchical control, i.e. the user can navigate across the available sub-dialogues.

It is well known that human conversants adapt various aspects of their conversational behavior to each other (cf. [34] for an overview). It is commonly accepted that also dialogue systems should adapt to their users. In order to achieve adaptivity in dialogue management we developed methods for online learning of policies for flexible interaction. We maintain dynamic state spaces that grow over time—during an interaction—according to the situations raised by the conversants. We extended our hierarchical reinforcement learning algorithm to support these dynamic states. We use our dialogue policy learning framework to infer the agent's behavior (or dialogue policy) from interactions with the environment.

The DM listens for events from the NLU and GRU components. It emits events to either the NLG or TTS component and to the NVBP component. For the purpose of experimenting with the overall system without relying on autonomous interpretation of speech and gesture input, and autonomous dialogue management, we developed a Wizard-of-Oz interface (Fig. 3). Given some user input to the robot, (e.g. "what is the correct answer"), the wizard selects the corresponding user dialogue act such as "Request(CorrectAnswer)". The DM then selects and executes the learned action by querying and updating the GMG and UM components. When the DM runs autonomously, it passes a dialogue act to the NLG and NVBP components. In a non-autonomous mode, the dialogue act selected by the DM is highlighted in the interface for the wizard to aprove or override. It is possible to switch between autonomous and wizarded DM at any time during a session with the system.

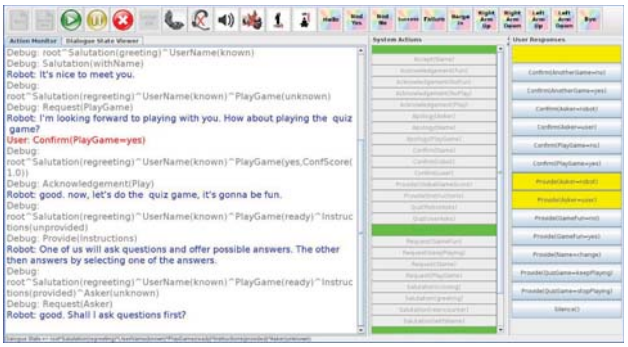

Figure 3: The Wizard-of-Oz GUI. The highlighted actions are the DM's suggestions to the wizard.

6. Spoken Output Processing

The communicative goal selected by the DM as the next system action needs to be verbalised and realised by speech, possibly accompanied by suitable non-verbal behaviour (e.g., waving with a greeting, nodding with positive feedback). To make the spoken output appealing, especially to children, we aim to avoid repetitive system output. We also developed a child-like voice for the robot, consistent with its role of a peer.

6.1. Natural Language Generation (NLG)

The communicative goal specifies the type of dialogue act and the values of a range of information state variables important for verbalisation selection. The range of dialogue acts in our system covers greetings and introductions, activity-management moves (e.g., request to play, request to switch roles, request of user turn, etc.), asking questions (e.g., engagement in a game or a quiz question), providing game move descriptions and instructions, providing information and comments on user's performance, various types of feedback and clarification requests. There are currently 60 types of communicative goals. Verbalisation is determined by an utterance planner using a set of graph rewriting rules. The output is either a string that is passed directly to the TTS, or a logical form that serves as input to a grammar-based lexical realisation component using the OpenCCG realizer [24] and the same handwritten grammar as mentioned above for parsing.

Long-term interaction involves series of encounters be-

tween the robot and a given user. In order to foster a sense of familiarity between them, the robot explicitly acknowledges and refers to common ground with a given user, thus making it explicit that it is familiar with the user. Examples of such verbalisations include: use of name (e.g., "So, which answer do you choose, Marco?"), references to previous encounters and play experiences (e.g., "I am happy to see you again"), references to previous performance in an activity (e.g., "You were again really good at the quiz game today"), reference to familiarity of a question in quiz or a dance move (e.g., "The next question should be familiar."), reference to the familiarity of activity rules (e.g., "Remember the magical pose?"). Such moves are accompanied by corresponding nonverbal behaviors, e.g., nodding, higher excitement.

It is well known that system output can be annoying and boring when the verbalisation of dialogue moves is repetitive. Since this could negatively influence engagement of children, we have invested considerable effort to implement a large range of verbal output variation. Selection among variants is either random or controlled by selectional criteria. Among the selectional criteria are various contextual parameters (e.g., the information whether the current user interacts with the system for the first time or it is a subsequent encounter, whether they have already played the current game or it is new, whether the user's previous performance was good or not) as well as characteristics of the content to be conveyed (e.g., whether a given quiz question was already asked or not, how many answer options a quiz question has and whether they are short or long, etc.). The resulting number of alternative verbalisations in the current implementation varies greatly between different dialogue acts, ranging from just a single verbalisation (for dialogue acts that only appear once, such as name introduction or the explanation of a game) to thousands of variants. This suffices to ensure that the users will not be exposed to repetitive system output.

A problem which often affects spoken dialogue system output is the lack of naturalness due to incorrect or ambiguous prosody of the generated sentence. We experiment with modifications of the spoken output prosody using the support for controling the prosody of TTS voices with symbolic markup of speech rate, pitch and contour. So far, two prosody modifications have been implemented:

- *Prosodic prominence modification (stress)* on words that realize the focus of a sentence: The NLG component labels focus words. In the current implementation, all occurrences of ordinal modifiers (e.g., 'seconda' in "ecco la seconda domanda" (here is the second question)) are labeled as focus, and thus to be stressed, because they denote a distinguishing property that delimits one of a set of possible alternatives in the sense of [35]. The TTS component then modifies the prosodic realization decreasing the speech rate and raising the pitch contour on the words that realize the focus.
- *Emotional prosody modification* according to the emotional state of the robot: Currently the dialogue manager decides when the system output should be rendered with (non-neutral) emotional coloring, either "sadly" or "happily". The TTS component implements a function `sayWithEmotion` that ensures the corresponding realization: increasing the speech rate and the pitch contour in the happy case, and decreasing them in the sad case.

6.2. Text-To-Speech Synthesis (TTS)

The commercial Acapela TTS system [36] is available by default on the Nao. However, the ALIZ-E project requires a more customizable and flexible TTS system: first, to support prosody modifications like those mentioned above to relate utterances to context and to express emotions vocally; second, to be able to use a child-like voice, because synthesized speech triggers social identification processes.

In order to achieve these objectives we chose the open source Mary TTS platform [37]. The advantage of using Mary TTS is that it supports state of the art technology in the field of HMM-synthesis [38], and enables us to experiment with the manipulation of para-verbal parameters (e.g. pitch shape, speech rate, voice intensity, pause durations) for the purpose of expressive speech synthesis, and the voice quality and timbre modifications algorithms [39] useful to to convert an adult TTS voice into a child like voice.

Mary TTS already supports many languages and it comes with a toolkit for quickly adding support for new languages and for building HMM-based synthesis voices. In order to develop a new Italian voice for Mary TTS, we first ported some of the existing Italian Festival TTS modules [40]: the basic NLP modules for Italian (Lexicon, Letter-To-Sound rules, Part Of Speech tagger) have already been ported into Mary TTS. A first test voice was created using the Wikipedia optimal text selection procedure and voice building procedure provided by Mary TTS. Informal listening tests yielded positive judgments.

7. Event-Based Component Integration

Due to the limited processing power and memory of the Nao's on-board computer, many of the system components must run on one or more PCs in the network. Moreover, we have pre-existing software written in different programming languages and running on different operating systems; therefore, a component integration framework is required.

The open source Urbi SDK [8] was chosen as the middleware in the ALIZ-E project. It aims at providing a universal programming environment orchestrating complex components. As a client-server architecture where the server is running on the robot, it is possible to integrate remote components written in C/C++ or Java with components that run directly on the robot. Urbi comes with a dedicated language, UrbiScript, which provides a number of interesting paradigms, for example for event-based and parallel programming, and can access and control the sensors and actuators of the Nao. Similar to other interactive systems, we had the choice between different component integration paradigms; the most popular ones appear to be publish-subscribe messaging [41] and blackboard [42] architectures. Essential requirements include proper encapsulation of components to ensure maintainability of the software; the flexible rearrangement of information flow; and a notification mechanism allowing a component to initiate the flow of information.

Urbi's event objects provide a suitable and robust mechanism for implementing a messaging paradigm: an Urbi event can carry arbitrary values as a 'payload'. Components can trigger events whenever new data is available, a certain processing stage has been reached, etc. A controller script, written in UrbiScript, implements event handlers which pass the information on to the appropriate components. The advantage of this approach is that all event handlers for a given instantiation of the system are maintained in a single file; beyond mere message passing, they can also provide additional functionality such as centralized logging.

One of the main problems in a robotic environment is how to deal with the need of having components that either (a) can access to low-level hardware details, (b) perform heavy compu-

tations, and typically run on different, more powerful machines, (c) should be coordinated concurrently, (d) should react to (typically asynchronous) events.

Languages such as C/C++ can suit low-level and heavy computational tasks well, but it can be tedious to manage concurrency, network communication and event handling with them. The Urbi environment provides the UrbiScript language which can orchestrate complex organisations of components named UObjects in highly concurrent settings. It is relatively easy to make a C/C++/Java program accessible as a UObject. Basically one needs to wrap the upstream program into a C++ (or Java) class inheriting from Urbi::UObject, then bind in urbiscript the methods that should be accessible from there. Eventually, one needs to define events and how to handle methods concurrently. We explain how this done using as example the integration of the ASR and TTS components into the system.

7.1. The ASR Component Integration

ASR is provided as an API whose functions can be accessed by other components (e.g., NLU). When ASR output is available, an event is launched and the result is provided as a payload, so that any component that needs this information can access it. The ASR component is basically made of two modules: a configuration structure (that holds data for AM and LM) and a main recognition loop function (also called "Julius stream"). The latter contains an internal VAD and outputs second pass recognition result as an NBest list. The principal methods of this component are: load/free/switch configuration; start/stop main recognition loop. Fig. 4 shows a scheme of function calls and data exchange among ASR, DM and NLU components.

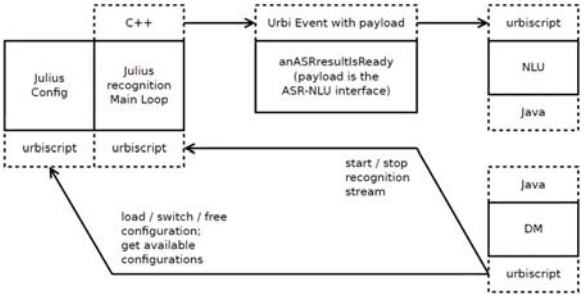

Figure 4: ASR communications through URBIscript.

Both speech recognition triggering and output notification are implemented by UrbiScript events. In particular, the N-best result is the payload carried to NLU component. We needed to create an Urbi data structure that could be populated by Julius C++ UObject output, and accessed by other components (for example the Java NLU). We implemented an N-Best interface for feeding the NLU component with ASR output. This data structure consists of a list of sentences; each of them holds total (i.e. sentence-wide) acoustic and linguistic probabilities. (Julius only outputs sentence-level acoustic and linguistic probabilities, and only a generic confidence score for word-level.)

7.2. The TTS Component Integration

We wanted to keep the possibility of using both the Acapela TTS and Mary TTS. We achieved this by implementing a configuration system able to choose between the different TTS systems. Acapela TTS is already available on Nao, but we needed to make Mary TTS available in the Nao/Urbi environment. Mary TTS is written in Java using the client/server paradigm. Due to the Nao CPU resources limitations, it has been decided to run the Mary TTS server on a remote PC while keeping the Acapela TTS as a built-in system on the Nao Robot. When Mary TTS produces the audio output it must be played from the Nao loudspeaker. This is achieved by using a streaming server based on gstreamer [43]. In order to have a real time interaction, an RTP [44] (Real-time Transport Protocol) streaming server is active on Nao. The incoming RTP stream is then connected to the robot's loudspeakers trough a gstreamer pipeline.

Although Urbi is able to manage RTP data, we follow the gstreamer approach, to avoid overloading the Urbi server. This approach also allows us to choose among many already available plugins for building audio/video pipelines. To bring Mary TTS and gstreamer RTP into the Urbi world, we created an UObject (UMaryTTS) as the principal UObject responsible for routing the synthesis request (Mary TTS client) and playing the resulting audio through different output channels. These channels are represented by the following UObjects:

- UMaryTTSAudioPlayer makes a request to the Mary TTS server and plays the resulting audio through the PC loudspeakers (useful for tests without a real robot).
- UMaryTTSRTPPlayer makes a request to the MaryTTS server and streams the resulting audio through an RTP connection using the efflux library [45].
- UMaryTTSGstreamerPlayer that makes a request to the MaryTTS server and streams the resulting audio through a UDP RTP permanent connection using gstreamer-java [43].

While Acapela TTS exposes only the functions `say(pText)` and `stopTalking()`, UMaryTTS also exposes `sayWithEmotion(pText, pEmotion)` that is able to change the global prosody settings according to a chosen emotion. Moreover UMaryTTS is able to manage the prosody labels assigned by the NLG component.

8. Experiments and Results

We apply a yearly cycle of specifications-development-experiments. Experiments with the integrated system with real users are carried out at the San Raffaele Hospital in Milan in order to assess the viability of the scenario, evaluate (certain aspects of) the system and collect data for further development. Experiments with the first version of the system were carried out in March 2011, and some results were discussed in [25]. Experiments with the new version were carried out in December 2011 (pilot) and in April–May 2012.

In these experiments, speech and gesture recognition and interpretation were fully wizarded (i.e., performed by a human, cf. Section 5), verbal and non-verbal output production was autonomous. Dialogue management was wizarded in the first weeks of the experiment, autonomous afterwards.

19 subjects aged 5–12 years (11 male, 8 female) participated. Every subject was invited to come three times to play with the robot. Each session took up to one hour. The children interacted on their own with the robot and then completed questionnaires gathering subjective evaluations. Time permitting, the child could select up to two games to play with Nao from the available repertoire of dance, imitation and quiz in each session. The child could stop an interaction at any time. The wizard could terminate an interaction too, for example due to technical problems or time constraints. The interactions playing one game with the system varied in length between 10–40

Evaluation Metric	Quiz wiz. DM	Quiz aut. DM	Imitation aut. DM
Dialogue Completion (%)	0.59	0.88	0.75
System Utterances x Dialogue	58.41	89.91	65.08
User Utterances x Dialogue	24.74	38.96	24.25
System Time x Utt. (secs)	8.13	6.27	8.29
User Time x Utt. (secs)	7.85	8.03	13.47
Autonomous System Actions (%)	0.27	0.94	0.96
Game Fun x Completed Dial. (%)	0.86	0.95	1.0

Table 2: Average results of objective metrics based on 51 dialogues for Quiz (27 with wizarded and 24 with autonomous DM) and 12 dialogues for Imitation with autonomous DM.

Experiment IDs	#Snt	#Wrd	WCR	Ins	WER
2011-12-13-pre00	4	22	77.3	31.8	54.5
2011-12-13-pre01	6	82	75.6	31.7	56.1
2011-12-13-pre02	5	40	80.0	17.5	37.5
2012-03-10-0020	7	63	74.6	3.2	28.6
2012-03-17-0021	15	114	90.4	4.4	14.0
2012-03-17-0022	4	49	59.2	8.2	49.0
2012-03-24-0021	12	107	62.6	5.6	43.0
2012-04-21-0026	11	84	67.9	8.3	40.5
Total	64	561	73.8	11.4	37.6

Table 3: Preliminary ASR results on quiz question recognition

minutes. Usually there was time to play about five games with a child over the three sessions. The total number of games played was 75. We collected video and separate audio recordings, as well as system log files. Analysis of this data has just started.

8.1. Objective Metrics

Table 2 gives the results of several objective metrics computed from the information saved automatically in the system log files during the quiz and imitation game interactions. We have log data for 51 quiz and 12 imitation game interactions. DM was wizarded in the first 27 quiz game interactions, autonomous in the remaining 24 and in all imitation interactions.

An interaction was counted as completed when it included the dialogue act Salutation(Closing). On the whole about 25% of interactions ended in a non-standard way, and thus as "not completed". This would typically happen due to a technical problem, e.g., the system locked up, and thus the interaction had to be aborted (this happenned more often in the early weeks, due to a technical problem).

The average number of system and user utterances only gives a very rough idea, because of big differences in the length of the interactions. But on the whole, the quiz system with autonomous DM appears more "talkative" than the wizarded one. The exact nature of this difference is to be analyzed. It is also worth noting that in all the interactions there are only 7 cases of the wizard typing some text to be synthesized as system response. This means that the available repertoire of dialogue moves generally sufficed to carry on the interaction, although we again need to investigate this in more detail. As for user utterances, the log files only register those for which the wizard entered some interpretation through the interface. There were utterances which the wizard "ignored", because they did not really influence the flow of the interaction. We are in the process of transcribing all user utterances, and then can evaluate this.

The system and user time per utterance includes both speaking and reaction time. Quiz interactions seem to be faster with autonomous DM. Imitation is slower than quiz due to the poses.

Before closing an interaction, the system asked the user whether it was fun playing the game, and we can see that the children mostly answered this question positively even when task success is moderate or weak; additional subjective evaluations were gathered through questionnaires and await analysis.

8.2. Spoken Input Interpretation

The data collected in the experiments can now be used for testing and further development of the component technologies. The ASR and NLU components are one such example.

As mentioned in Section 4.1.3 we built a specific LM for the questions and answers in the Quiz Database. Table 3 shows the results of ASR applied to 64 utterances (561 words) where a user poses a quiz question to the robot. On average, we get 74% correct words, 11.5% inserted words and 38% WER.

Taking the ASR hypotheses as input to the NLU, 56 questions were correctly identified by fuzzy matching against the quiz database contents (Section 4.2). This is an encouraging first result and further experiments will show whether this level of ASR+NLU performance suffices to sustain the interaction. Further transcriptions of the experiment data are ongoing, which will enable more evaluation.

9. Conclusions and Outlook

We presented a human-robot interaction system that recognizes input and produces output in the form of speech and gestures. We described the components developed to build the system, with focus on spoken input recognition and interpretation, spoken output production and dialogue management. The components are integrated using an event-based approach implemented in the Urbi middleware. This approach supports integration of written in different programming languages, running in parallel, distributed on several computers. The components exchange information by values carried as 'payload' by Urbi events which they trigger whenever new data is available, a certain processing stage has been reached, etc. This allows us to create a flexible processing model.

We discussed some specifics of building a system for children about 8–11 years old, and how we address these challenges. This involves building acoustic and language models especially for this purpose, developing robust natural language interpretation, flexible and adaptive dialogue management, varied spoken output production and expressive synthesis with a child-like voice. The system has been evaluated in experiments with target users. On the one hand, the experiments confirmed the viability of our scenario using a system with wizarded input interpretation. On the other hand, we collected data for further development. Initial evaluation of the spoken input interpretation on a small subset of the data yield encouraging results.

The immediate future steps are clear. One the one hand, more component-level evaluation using the collected data. For example, to test spoken input recognition and interpretation on other utterances than users posing quiz questions to the system. On the other hand, more analysis of the data, for example to determine areas of improvement for the dialogue manager.

10. Acknowledgements

Parts of the research reported on in this paper were performed in the context of the EU-FP7 project ALIZ-E (ICT-248116) [4].

11. References

[1] A. Tartaro and J. Cassell, "Using virtual peer technology as an intervention for children with autism," in *Universal Usability: Designing Computer Interfaces for Diverse User Populations*. New York, John Wiley & Sons, Ltd., 2006, pp. 231–262.

[2] S. N. Shrikanth and A. Potamianos, "Creating conversational interfaces for children," *IEEE Trans. SAP, Vol. 10, No. 2*, pp. 65–78, 2002.

[3] S. Yildirim, S. S. Narayanan, and A. Potamianos, "Detecting emotional state of a child in a conversational computer game," *Computer Speech and Language, Vol. 10, No. 1*, pp. 29–44, 2011.

[4] "ALIZ-E website," (accessed 10.5.2012), http://aliz-e.org/.

[5] O. Lemon, A. Bracy, A. Gruenstein, and S. Peters, "The WITAS multi-modal dialogue system I," in *Proc. Eurospeech'01*, Aalborg, Denmark, Sep 2001, pp. 1559–1562.

[6] R. Stiefelhagen, H. Ekenel, C. Fugen, P. Gieselmann, H. Holzapfel, F. Kraft, K. Nickel, M. Voit, and A. Waibel, "Enabling multimodal human-robot interaction for the Karlsruhe humanoid robot," vol. 23, no. 5, 2007, pp. 840–851.

[7] I. Kruijff-Korbayov, G. Athanasopoulos, A. Beck, P. Cosi, H. Cuayhuitl, T. Dekens, V. Enescu, A. Hiolle, B. Kiefer, H. Sahli, M. Schrder, G. Sommavilla, F. Tesser, and W. Verhelst, "An event-based conversational system for the nao robot," in *Proc. IWSDS'11*, Granada, Spain, Sep 2011.

[8] J. Baillie, "Urbi: Towards a universal robotic low-level programming language," in *2005 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. IEEE, 2005, pp. 3219–3224.

[9] A. Beck, L. Cañamero, and K. Bard, "Towards an affect space for robots to display emotional body language," in *Proc. Ro-Man'10*, ser. Ro-Man 2010. IEEE, 2010, pp. 464–469.

[10] T. Dekens and W. Verhelst, "On the noise robustness of voice activity detection algorithms," in *Proc. Interspeech'11*. ISCA, Sep 2011, pp. 2649 – 2652.

[11] G. Athanasopoulos, H. Brouckxon, and W. Verhelst, "Sound source localization for real-world humanoid robots," in *Proc. SIP'12*. WSEAS, Mar 2012, pp. 131 – 136.

[12] G. Athanasopoulos, T. Dekens, H. Brouckxon, and W. Verhelst, "The effect of speech denoising algorithms on sound source localization for humanoid robots," in *Proc. ISSPA'13*, July 2012.

[13] M. Gerosa, D. Giuliani, S. S. Narayanan, and A. Potamianos, "A review of asr technologies for childrens speech," *Proc. WOCCI'09*, vol. 49, pp. 847–860, Feb 2009.

[14] A. Potamianos and S. Narayanan, "Robust recognition of children's speech," *IEEE Trans. SAP, Vol. 11, No. 6*, pp. 603–616, 2003.

[15] K.-F. Lee, H.-W. Hon, and R. Reddy, "An overview of the sphinx speech recognition system," *IEEE Trans. ASSP*, vol. 38, no. 1, pp. 35 – 45, Jan 1990.

[16] "Julius development team. Open-Source Large Vocabulary CSR Engine Julius." (accessed 10.5.2012), http://julius.sourceforge.jp/.

[17] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland, "The htk book, version 3.4," in *Cambridge, UK: Cambridge University Engineering Department*, 2006, http://julius.sourceforge.jp/.

[18] "VoxForge. Tutorial: Create Acoustic Model - Manually." (accessed 10.5.2012), http://www.voxforge.org/home/dev/acousticmodels/linux/create/htkjulius/tutorial.

[19] "EVALITA 201: Forced Alignment Task. Tech. Report 2012 (Francesco Cutugno et al.)," (accessed 10.5.2012), http://www.evalita.it/sites/evalita.fbk.eu/files/working_notes2011/Forced_Alignment/FORCED_ORGANIZERS.pdf.

[20] M. Gerosa, D. Giuliani, and F. Brugnara, "Acoustic variability and automatic recognition of children's speech," *Speech Communication*, vol. 49, pp. 847–860, Feb 2007.

[21] M. Nicolao and P. Cosi, "Comparing SPHINX vs. SONIC Italian Children Speech Recognition Systems," in *Proc AISV'11*, Feb 2011, pp. 414–425.

[22] A. Stolcke, "Srilm - an extensible language modeling toolkit," 2002, pp. 901–904.

[23] J. Baldridge and G.-J. Kruijff, "Multi-modal combinatory categorial grammar," in *Proc. EACL'03*, 2003.

[24] "OpenCCG website," (accessed 10.5.2012), http://openccg.sourceforge.net/.

[25] R. Ros, M. Nalin, R. Wood, P. Baxter, R. Looije, and Y. Demiris, "Child-Robot Interaction in The Wild: Advice to the Aspiring Experimenter." in *Proc. ICMI'11*, Valencia, Spain, Nov 2011, pp. 335–342.

[26] N. Roy, J. Pineau, and S. Thrun, "Spoken dialogue management using probabilistic reasoning," in *International Conference on Computational Linguistics (ACL)*, Hong Kong, Oct 2000, pp. 93–100.

[27] J. Williams and S. Young, "Partially observable Markov decision processes for spoken dialog systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 393–422, 2007.

[28] B. Thomson, "Statistical methods for spoken dialogue management," Ph.D. dissertation, University of Cambridge, 2009.

[29] Y. Young, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, T. B., and K. Yu, "The hidden information state model: a practical framework for pomdp-based spoken dialogue management," *Computer Speech and Language*, vol. 24, no. 2, pp. 150–174, 2010.

[30] M. Frampton and O. Lemon, "Recent research advances in reinforcement learning in spoken dialogue systems," *Knowledge Engineering Review*, vol. 24, no. 4, pp. 375–408, 2009.

[31] H. Cuayáhuitl, S. Renals, O. Lemon, and H. Shimodaira, "Evaluation of a hierarchical reinforcement learning spoken dialogue system," *Computer Speech and Language*, vol. 24, no. 2, pp. 395–429, 2010.

[32] H. Cuayáhuitl, "Learning dialogue agents with bayesian relational state representations," in *Proc. IJCAI Workshop KRPDS'11*, Barcelona, Spain, Jul 2011, pp. 9–15.

[33] H. Cuayáhuitl and I. Kruijff-Korbayová, "An interactive humanoid robot exhibiting flexible sub-dialogues." in *Proc. NAACL-HLT*, Montreal, Canada, Jun 2012, (to be published).

[34] S. Oviatt, C. Darves, and R. Coulston, "Toward adaptive conversational interfaces: Modeling speech convergence with animated personas," *ACM Trans. Comput.-Hum. Interact.*, vol. 11, no. 3, pp. 300–328, Sep. 2004. [Online]. Available: http://dx.doi.org/10.1145/1017494.1017498

[35] M. Steedman, "Information structure and the syntax-phonology interface," *Linguistic Inquiry*, vol. 31, no. 4, pp. 649–689, 2000.

[36] "Acapela website," (accessed 10.5.2012), http://www.acapela-group.com/index.html.

[37] "Mary TTS website," (accessed 10.5.2012), http://mary.dfki.de/.

[38] H. Zen, T. Nose, J. Yamagishi, S. Sako, T. Masuko, A. Black, and K. Tokuda, "The HMM-based speech synthesis system (HTS) version 2.0," in *Proc. ISCA SSW6*, 2007, pp. 294–299.

[39] F. Tesser, E. Zovato, M. Nicolao, and P. Cosi, "Two Vocoder Techniques for Neutral to Emotional Timbre Conversion," in *7th Speech Synthesis Workshop (SSW)*, Yoshinori Sagisaka and K. Tokuda, Eds. Kyoto, Japan: ISCA, 2010, pp. 130–135.

[40] P. Cosi, F. Tesser, R. Gretter, C. Avesani, and M. W. Macon, "Festival speaks italian!" in *Proc. Interspeech'01*, 2001, pp. 509–512.

[41] M. Schröder, "The SEMAINE API: towards a standards-based framework for building emotion-oriented systems," *Advances in Human-Computer Interaction*, vol. 2010, no. 319406, 2010.

[42] N. Hawes, J. L. Wyatt, A. Sloman, M. Sridharan, R. Dearden, H. Jacobsson, and G.-J. Kruijff, "Architecture and representations," pp. 53–95, 2009.

[43] "gstreamer-java development team. Java interface to the gstreamer framework." (accessed 10.5.2012), http://code.google.com/p/gstreamerjava/.

[44] "Wikipedia. Real-time Transport Protocol." (accessed 10.5.2012), http://en.wikipedia.org/wiki/Real-time_Transport_Protocol/.

[45] "efflux development team. efflux: a Java RTP stack with RTCP support and a clean API." (accessed 10.5.2012), https://github.com/brunodecarvalho/efflux/.