An Integrated Framework for Multimodal Human-Robot Interaction

Luis Fernando D'Haro*, Andreea I. Niculescu*, Caixia Cai†, Suraj Nair†,
Rafael E. Banchs*, Alois Knoll§ and Haizhou Li*

* Human Language Technology Dept. Institute for Infocomm Research, A*STAR
One Fusionopolis Way # 21-01, Connexis South Tower, Singapore, 138632
E-mail: {luisdhe, andreea-n, rembanchs, hli}@i2r.a-star.edu.sg Tel: +65-6408 2146

† TUMCREATE, One Create Way #10-02 Create Tower, Singapore 138602
E-mail: {caixia.cai, suraj.nair}@tum-create.edu.sg Tel: +65 6601 4016

§ Robotics and Embedded Systems, Technische Universität München, 80333, Germany
E-mail: {knoll}@in.tum.de Tel: +49 89289 18104

***Abstract*— Recent research progresses in speech recognition, text-to-speech, natural language understanding, or dialog management components are improving the way humans interact with advanced robot machines. However, far from being solved, we are just starting the process of creating meaningful multimodal platforms that can allow operators to use and control industrial robots through spoken dialogue.**

This paper describes our ongoing efforts on creating a modular platform that combines different technologies to cover typical requirements in an industrial setting, i.e. robust speech recognition, low level skill functions to operate the robot, recommendations and validation procedures to setup parameters, combination of audio-visual information for challenging environments, integration of domain-knowledge by means of an ontology, a flexible definition of the dialog model and natural language rules, as well as a test and control interface to quickly check the functionality of each module during development and operation. All platform modules are intercommunicated by the ROS operative system which allows the integration of external plugins and modules easily.

Finally, a preliminary user study with IT experts simulating a welding task has been doing giving us clues on what should be the focus of our next developments.

I. INTRODUCTION

Human-Robot interaction is an important topic that is gaining more and more attention as many robust and highly accurate technologies (e.g. speech and object recognition, natural language understanding, grounding, etc.) are included into commercial or industrial robots in order to allow entertainment or the performance of repetitive tasks. In this kind of setups, speech is a natural modality for communication. However, its integration with other components and modalities poses some requirements in terms of reliability, speed, flexibility and modularity. In this paper, we want to describe a scalable platform which allows different kind of industrial tasks to be performed (in our case, we have used to resemble a welding task, a gluing task and a drawing by example task [1][2]). This way, for instance the operator can set the values of different parameters required for each task, e.g. the speed of the welding process, the name of the piece to glue, request default values or recommended values for the task to perform, or to perform some frame-based dialogs. Since providing detailed information about each of the tasks we implemented is beyond the scope of this paper, we prefer instead to explain the whole architecture and detailed information about each module and its implementation.

The paper is arranged as follows: section II fully describes each module in the architecture of the proposed system. Section III describes the usability evaluation done with experts; finally, section IV presents the conclusions and future work.

II. PROPOSED ARCHITECTURE

This section describes each of the integrating modules of the platform (see Figure 1), their capabilities and limitations.

A. Speech Recognizer and Machine Translation

The main goal for the Automatic Speech Recognizer (ASR) is to transcribe the uttered speech by the operator when requesting the robot or the system to perform a given action. Currently there are two main approaches applied to speech to text transcription systems: a) continuous Hidden Markov Models (HMM) [3], and b) deep neural networks (DNN) models [4]. Recent improvements in the latter have allowed achieving results that are even better than human transcribers [5]. Given that the ASR provides one of the input modalities used by the operator, the requirements for a high quality transcription, noise and channel robustness, as well as the easiness to integrate it into the developed interface (see section I) are important parameters to select the ASR.

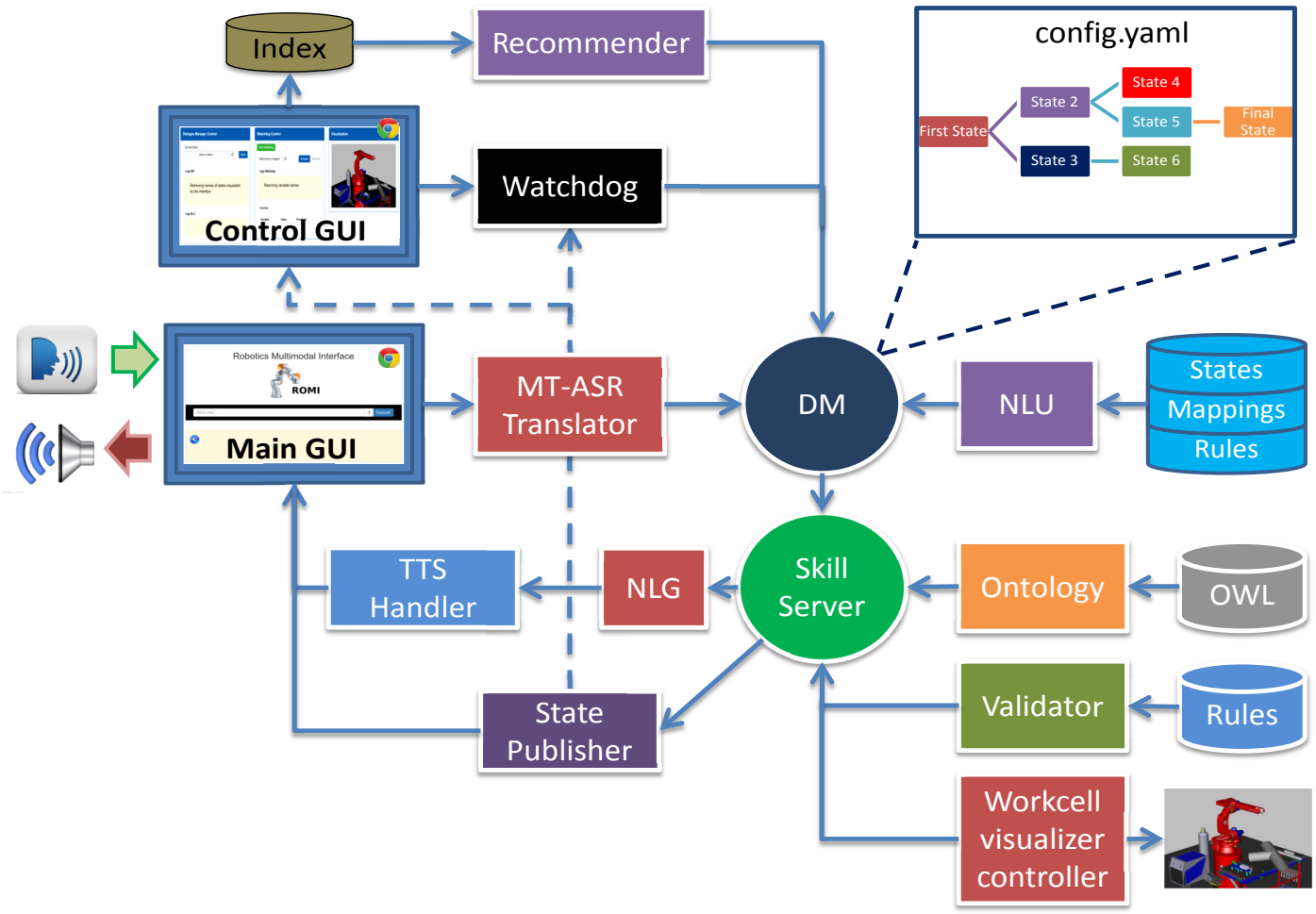

Figure 1: S*ystem architecture*

In [6] a comparative study among six different recognizers is presented. Overall, along different domains and audio conditions, one of the best recognizers was Google ASR, therefore we decided to use it in our system as default recognizer (although other engines can be integrated). The Google ASR is integrated in the Chrome browser by using the webkitSpeechRecognition toolkit, which connects with their cloud based service and sends the collected speech to the service as a HTML POST request and receive back a sorted n-best list of transcription results.

Despite all its advantages, it has an important limitation since developers cannot adapt the acoustic models, and there are limitations to specify language models and vocabulary. In [7] we proposed an automatic correction mechanism based on using a machine translation system trained using words and phonetic encoding representations of the n-best lists of ASR results. This mechanism provides a quick but robust mechanism to improve the ASR performance by reducing the word error rate (WER), the occurrence of OOVs, and increasing the matching of the corrected transcriptions to the ones allowed by the NLU grammars. In our system, after each recognition, the interface calls the MT-ASR service which retrieves the translated/adapted result.

B. Text-to-Speech and Natural Language Generation

The goal of the Natural Language Generator (NLG) is to define how the concepts coming from different modules in the architecture are expressed to the operator in the form of syntactic structures and words. Two main approaches are addressed in the literature: template-based (or prompts) and generative models [8]. Our current implementation allows mainly the usage of templates defined in the definition of the dialog states (see section D), although basic generation is also allowed by using the open-sourced toolkit SimpleNLG[1].

The Text-To-Speech (TTS) converts a given prompt or generated sentence into an acoustical signal that can be played to the user. Several high quality TTS are available specially based on unit selection synthesis [9], although there are recent advances using deep learning with important improvements in quality [10] and speed [11]. Our current module makes use of Google TTS voices (using the speechSynthesis library available in Chrome) which allows the selection of up to 18 different voices in different languages (English, Spanish, Chinese, Russian, etc.). Designers can change the default voice easily by using the control interface (see section J).

[1] https://github.com/simplenlg/simplenlg

C. *Natural Language Understanding*

The NLU provides an interpretation of a given sentence or command (e.g. typed in on a textbox in the GUI or recognized using the ASR) in a form that can be used by the computer. Several approaches for parsing have been proposed in the literature from using semantic grammars [12], hidden Markov models [13], machine translation approaches [14], conditional random fields [15], or deep learning networks [16][17]

Our NLU module is based on the use of regular expressions intended to match the given sentence with a set of predefined patterns. The use of regular expressions provides a trade-off between maintenance, available resources, accuracy, diversity of the command sentences, and robustness which are very common and important factors in industrial settings. Developers can specify the set of rules that are active for each state with the goal of increasing the accuracy and speeding up the parsing process. Besides, it is possible to specify a set of general rules that are always active independent of the state (e.g. salutations, yes/no, help requests, etc.). This capability of the system to switch between general and specific rules allows the system to deal with the dynamic characteristics of the human-computer interactions, while reducing the designer workload. Finally, the module includes the possibility of creating one-to-many mappings allowing the creation of regular expressions using special labels (e.g. NUMBERS) which are expanded at real time using the OR condition in the regular expression.

D. *Dialog Management (DM)*

The DM is the most complex module since it must handle asynchronously different sources of information and commands (e.g. the NLU interpretation of the operator speech, the direct inputs from the interface), as well as interchanging information with other modules (skills server, watchdog, NLG), everything to be done in a correct and prioritized sequence of actions to successfully achieve the required tasks. In the literature we can find different approaches to the definition of the actions (i.e. through state machines, frames, or reinforcement learning techniques [18][19]), as well as toolkits like OpenDial [20], Galaxy [21], or Trindikit [22].

Currently, our DM module follows the states and transitions defined in a state machine (specified in a configuration file as shown in Figure 2); here the designer can specify the default values for process or internal variables (number 1 and 2 in the figure), the different states (number 3), prompts that the system will use to call the attention of the user to request or provide him/her a given information, as well as possible conditions for each prompt (4), e.g. using different prompts depending on the times the state is trigger, or to provide help prompts to make clear to the user what s/he needs to say (5), and the actions to be done by the skill server, calls to ROS services and input arguments, and the conditions to jump to another state, or in case of error (6 and 7).

To speed up the design, there are some predefined actions done by the DM without the designer being required to specify them (although some level of configuration is allowed). For instance, the DM classifies the messages coming from different modules in the robot and notifies the user about the actions to be done based on them. This way, if the NLU cannot extract meaningful information from an utterance, then the DM sends a message to the NLG/TTS to inform the user that s/he needs to repeat the utterance again. Or if the validation of the parameters is unsuccessful, the DM receives the message and sends it to the interface to inform the user.

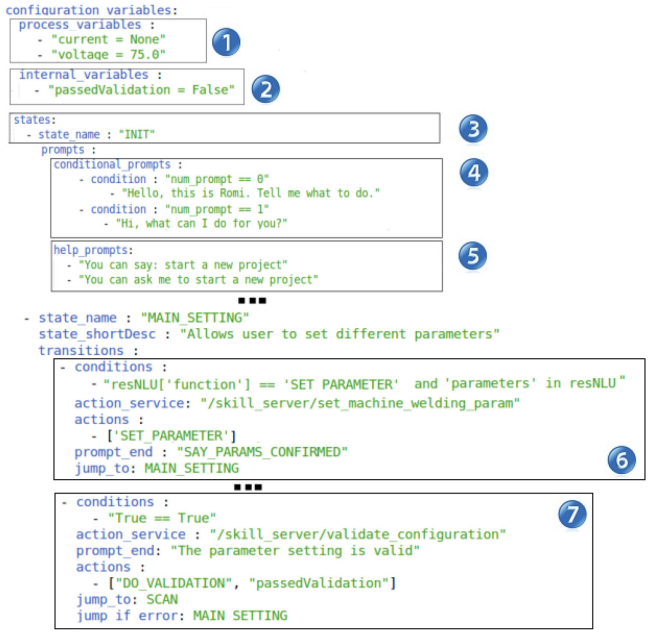

Figure 2. YAML configuration file for defining states

E. *Skill Server*

This module acts as a mediator between the low level primitives of the simulator, the validator, or the access to the ontology providing a high-level API for other modules. This way, specific details of the robot implementation are handled while keeping the same calls from the platform point of view.

F. *Ontology and State Publisher*

The ontology is the formal representation of the types, properties and relationships between the entities that can be found in the domain of the particular task. For example, this information will allow a welder robot to know the characteristics of the pieces to weld or the best parameters and available tools to perform the task. The ontology can also be used to save or access general information that is available to any module in the platform. For our welding application, the ontology (OWL and RDF content) was created using Protégé and stored on a SESAME repository running in a Tomcat server and accessible by means of a RESTful-based interface.

On the other hand, the state publisher is a ROS publisher that gets all the parameters from the ontology and frequently publishes them as a topic to any module subscribed to the topic. In addition, the module provides an ordered way for all other modules to read and write information into the ontology. In our system, the state publisher is accessed mainly by the

web interface or the DM every time the user changes the value of one of the task parameters by typing or using the speech, or whenever the user loads a previous configuration project; in addition, the validator and recommender modules use it to know the limits or thresholds of the task parameters.

G. Validation and Recommendation

The goal of the validation module is to check that the parameters provided by the user are correct and that the robot has what is required to perform the task (e.g. the robot is operative or there are enough source materials); The validator consults the ontology by means of a set of expert rules and in case of an error, it will send a warning or error message to the user and forbid the execution of the task.

On the other hand, the recommendation module allows expert knowledge to be integrated into the system which can provide default parameters for specific or common situations (e.g. valid ranges of parameters to perform the welding of a given piece). Our current module uses an index allowing the designer to provide sets of question-answer pairs (i.e. FAQ).

H. Watchdog

This module is responsible for doing a periodical scanning or checking of the "robot health" in order to guarantee that it can perform the tasks required by the user. This checking is inspired in the human autonomic nervous system (ANS), where critical tasks are performed independently, and with higher priority, over the somatic nervous system (SNS) in order to guarantee the correct functionality of the body. In the robot, the SNS tasks typically are the ones required by the human operators through the multimodal interface (e.g. performing the welding, scanning pieces to weld, etc).

While the ANS tasks mainly refer to periodically checking the robot internal functionalities, safety regulations or availability of source materials (e.g. power status, temperature, close presence of humans, availability of welding material, etc); the module provides visual and audible alerts to the operator, and perform actions like stopping the robot in case of problems.

I. Multimodal Graphical User Interface

The web-based interface (Figure 3) allows the operator to get access to several configuration parameters required to perform the task. The interface was designed following requirements found during our field study in [23] and implemented using HTML5 and CSS3 specifications, modular Javascript libraries, including the Rosbridge[2] library, to allow the communication with ROS and the available topics, services and publishers, as well as handling compatibility among browsers and operative systems

In addition, we used the MpegCanvasJS[3] library to stream the simulator images into the browser, and made use of responsive libraries like Booststrap to allow displaying on different mobile or desktop screens, and the Validator library[4] to allow the connection to our modules to quickly validate the parameters introduced through the forms in the screen.

J. Control Interface

Given the amount of different modules that are available in the platform and the number of configurable parameters that each of them could have (e.g. grammars, states, speed of the robot, thresholds for alerts, etc.), we created a web interface (Figure 4) that could allow developers to start/stop each module, manually modify its configuration parameters, test the output of the most common services for each module, and check the logs of each module independently.

Here, we used the Rosbridge library again to allow the communication with the different modules; however, in order to allow starting or stopping each module (or all of them), we implemented a post request service into our web server which receives the commands from the interface by using AJAX messages and then executes the required ROS launch files, services or kill functions. This workaround was needed since Rosbridge currently do not implement this functionality.

K. Communication Framework (ROS)

The Robot Operating System (ROS) is a robotics middleware consisting of a huge collection of software libraries and tools that help developers to build robot applications providing operating system-like functionality on heterogeneous environments, as well as low-level device control, hardware abstraction, message-passing between processes, package management, and open-sourced implementations of commonly used functionalities. Since all our modules make use of ROS, it is possible to include new plugins and third-party components to expand the capabilities of the platform.

L. Simulator

In order to evaluate the different tasks that the real robot can perform, we included a robot simulator which resembles a COMAU arm and includes different tools (e.g. gripper, scan sensor, welding gun, etc.) that can be interchanged depending on the required task. The code is based on the rlCoachMdl demo of the robotics library[5], but it can be replaced for another one provided that the designer modifies the calls and parameters defined in the configuration file (see section D).

III. Evaluation

Given the current integration and development stage of our platform, we focused on evaluating it with IT experts who were able to provide quick feedback on the integrated technologies and overall performance, welder's challenges and preparation techniques needed to ensure a good welding quality.

Our evaluation was carried out using both qualitative and quantitative methods: the qualitative study was performed with 4 participants using Nielsen's heuristics in a think-loud approach.

[2] http://wiki.ros.org/rosbridge_suite

[3] http://wiki.ros.org/mjpegcanvasjs

[4] https://github.com/1000hz/bootstrap-validator

[5] http://www.roboticslibrary.org/

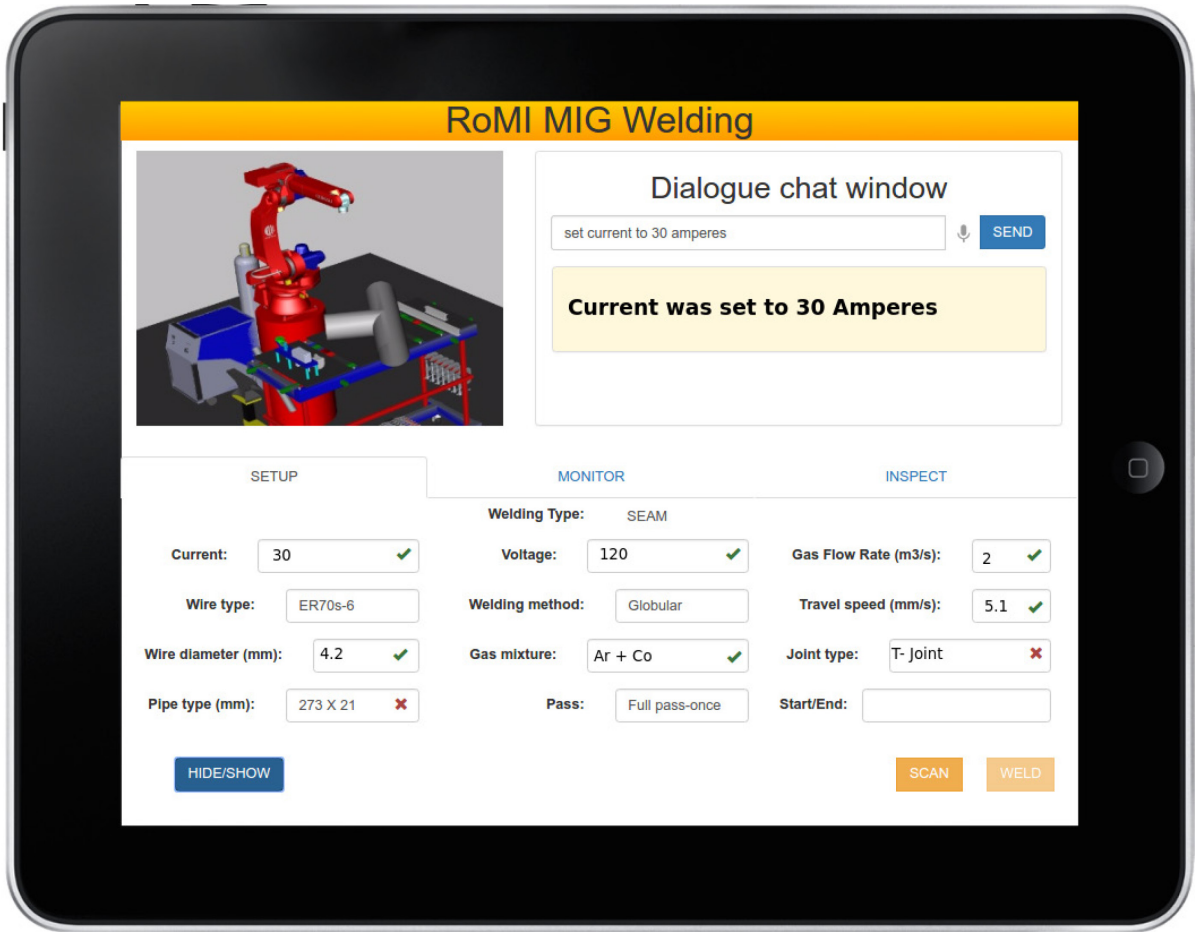

Figure 3. Aspect of the main graphical interface

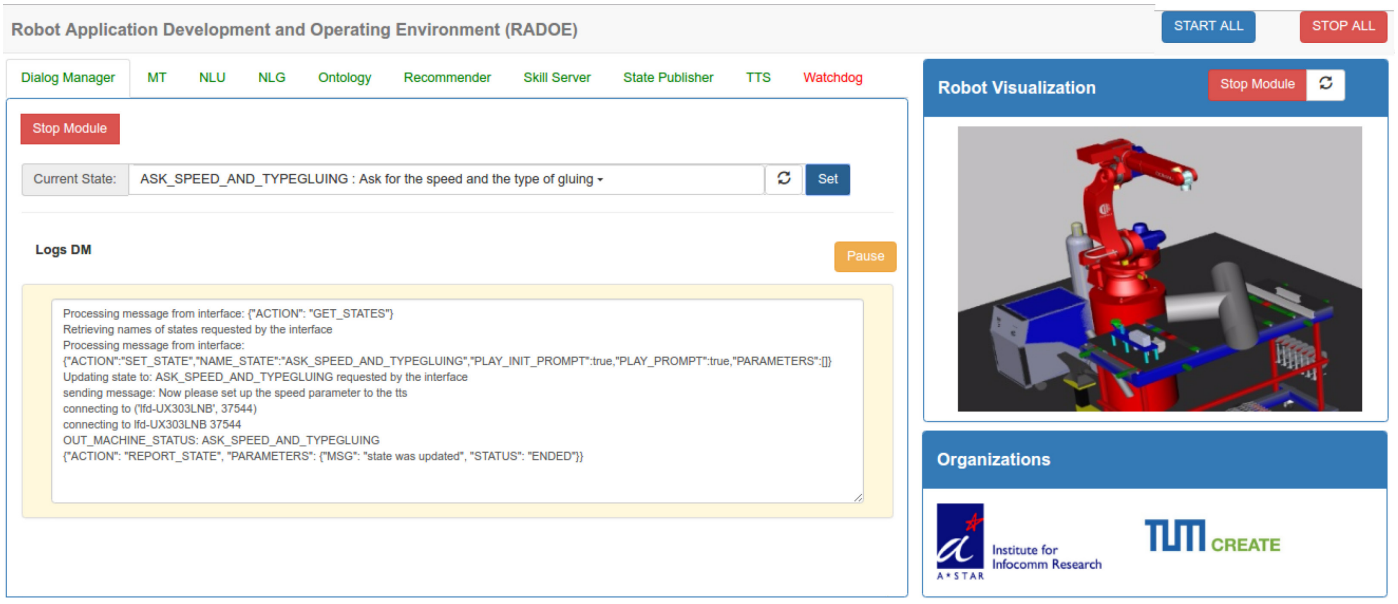

Figure 4. Interface to test and configure each module

In the second study, 15 participants performed 3 scenarios and evaluated the interface using a questionnaire focusing on screen layout, terminology & system information, multimodal combination, system capabilities, voice quality, verbosity and overall user experience. For both studies participants were briefed before the experiment. The briefing consisted in watching a power point presentation and a video about presenting a user case welding scenario. Results from both studies showed that the user interface d design was found to be pleasant, clear and task supportive. The speech functionality was found to be very useful for hands free interaction, feedback and Q&A session. Generally, participants seemed to appreciate the multimodal interaction features. The camera streaming was found to be a must-have functionality. Participants also requested a zoom in/out feature. Improvement suggestions referred to the recognition robustness, understanding capabilities, support for error recovery and decreasing information redundancy. More information about the evolution can be found in [24].

IV. Conclusions And Future Work

This paper described in detail an integrated platform for operating and controlling industrial robots by using speech and a task oriented graphical interface. The proposed architecture consists of independent modules that communicate among them by means of the ROS operative system which also allows the integration of external components. In addition, a control interface and a defined set of services and topics allow developers to configure, replace or extend the provided modules, as well as re-train and update MT models and NLU rules. Finally, results of a survey with IT experts allowed us to evaluate the usability of the proposed platform and plan for new features and improvements.

As future work, we plan to improve the robustness of the ASR module by a) allowing developers to define domain-state dependent language models that can be passed to the ASR to restrict the vocabulary of the transcriptions and avoid the occurrences of OOV terms, and b) including automatic mechanisms that can build the MT models without necessarily requiring users to train it with their own speech [25]. Besides, we are considering the possibility of creating a graphical interface (similar to [26] or [27]) that can be used to define and test the dialog state machine, prompts, transition conditions, and grammars defined in the configuration file. Also, we plan to include new speech and NLP modules from our internal platform [28][29] by porting them to ROS framework. Finally, we will be introducing additional strategies for error handling as proposed in [30].

Acknowledgment

This project was supported by SERC Industrial Project (EC-2013-045). We also thank Aravindkumar Vijayalingam and Eloy Retamino for their contributions to the project.

References

[1] Wu, Y., Chan, W. L., Li, Y., Tee, K.P., Yan, R., and Limbu, D.K., "Improving Human-Robot Interactivity for Tele-operated Industrial and Service Robot Applications", i*n Proceedings of the 7th IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM)*, July 2015, Cambodia.

[2] Ko, W.K.H., Wu, Y., and Tee, K.P., "LAP: A Human-in-the-loop Adaptation Approach for Industrial Robots", *in Proceedings 4th International Conference on Human-Agent Interaction (HAI 2016)*, Oct 2016, Singapore.

[3] Young, S. “A review of large-vocabulary continuous-speech”. *Signal Processing Magazine*, IEEE, 13(5), 45. 1996.

[4] Hinton, G., Deng, L., Yu, D., Dahl, G. E., Mohamed, A. R., Jaitly, N., & Kingsbury, B. “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups”. *Signal Processing Magazine, IEEE*, 29(6), 82-97. 2012.

[5] Xiong, Wayne, Jasha Droppo, Xuedong Huang, Frank Seide, Mike Seltzer, Andreas Stolcke, Dong Yu, and Geoffrey Zweig. "Achieving human parity in conversational speech recognition." *arXiv preprint* arXiv:1610.05256 (2016).

[6] Morbini, F., Audhkhasi, K., Sagae, K., Artstein, R., Can, D., Georgiou, P., & Traum, D. “Which ASR should I choose for my dialogue system?” *Proc. SIGDIAL*, August, 2013.

[7] D’Haro, L. F., Banchs, R. E. “Automatic Correction of ASR Outputs by Using Machine Translation”, *in proceedings Interspeech 2016*, 3469-3473. 2016

[8] Reiter, Ehud, Robert Dale, and Zhiwei Feng. “Building natural language generation systems”. *Vol. 33. Cambridge*: Cambridge university press, 2000.

[9] Hunt, Andrew J., and Alan W. Black. "Unit selection in a concatenative speech synthesis system using a large speech database." *In Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, 1996. ICASSP-96., vol. 1, pp. 373-376. IEEE, 1996.

[10] Aaron van den Oord, Sander Dieleman, Heiga Zen, et al. “WaveNet: A Generative Model for Raw Audio”, *arXiv preprint* arXiv:1609.03499. 2016.

[11] Sercan O. Arik, Mike Chrzanowski, Adam Coates, et al. “Deep Voice: Real-time Neural Text-to-Speech”, *arXiv preprint* arXiv: 1702.07825. 2017.

[12] Issar, Sunil, and Wayne Ward. "CMU's robust spoken language understanding system." *In Proceedings of Eurospeech*, vol. 93. 1993.

[13] Pieraccini, Roberto, Esther Levin, and Chin-Hui Lee. "Stochastic Representation of Conceptual Structure in the ATIS Task." *In HLT*. 1991.

[14] Macherey, Klaus, Franz Josef Och, and Hermann Ney. "Natural language understanding using statistical machine translation." *In INTERSPEECH*, pp. 2205-2208. 2001.

[15] Yao, Kaisheng, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao. "Recurrent conditional random field for language understanding." *in Proceedings IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2014 pp. 4077-4081. IEEE, 2014.

[16] Sarikaya, R., Hinton, G. E., & Deoras, A. “Application of deep belief networks for natural language understanding”, *in IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 22(4), 778-784. 2014.

[17] Jaech, Aaron, Larry Heck, and Mari Ostendorf. "Domain adaptation of recurrent neural networks for natural language understanding." *arXiv preprint* arXiv:1604.00117 (2016).

[18] Levin, Esther, Roberto Pieraccini, and Wieland Eckert. "A stochastic model of human-machine interaction for learning dialog strategies." *IEEE Transactions on speech and audio processing* 8, no. 1 (2000): 11-23.
[19] Paek, Tim, and Roberto Pieraccini. "Automating spoken dialogue management design using machine learning: An industry perspective." *Speech communication 50*, no. 8 (2008): 716-729.
[20] Lison, Pierre, and Casey Kennington. "OpenDial: A toolkit for developing spoken dialogue systems with probabilistic rules." *ACL 2016* (2016): 67.
[21] Seneff, Stephanie, Edward Hurley, Raymond Lau, Christine Pao, Philipp Schmid, and Victor Zue. "GALAXY-II: a reference architecture for conversational system development." In *ICSLP*, vol. 98, pp. 931-934. 1998.
[22] Ljunglöf, Peter. "trindikit.py: An open-source Python library for developing ISU-based dialogue systems." *Proc. of IWSDS* 9 (2009).
[23] Niculescu, Andreea I and D'Haro, Luis Fernando and Banchs, Rafael E and Yeo, Kheng Hui and Vyas, Dhaval. "Understanding welding practices on shipyards: An ethnographic study for designing any interactive robot welder" *in Proceedings 3rd International Conference on User Science and Engineering (i-USEr)*, 2014.
[24] Niculescu, Andreea. I., Luis Fernando D'Haro and Rafael E. Banchs, "When industrial robots become more social: on the desihn and evaluation of a multimodal interface for welding robots," APSIPA 2017, Kuala Lumpur, Malaysia.
[25] D'Haro, Luis Fernando, Seokhwan Kim, and Rafael E. Banchs. "A robust spoken Q&A system with scarce in-domain resources." *In Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2015 Asia-Pacific, pp. 47-53. IEEE, 2015.
[26] McTear, Michael F. "Developing a Spoken Dialogue System Using the CSLU Toolkit", *Spoken Dialogue Technology*, pp. 163-203, Springer London, 2004.
[27] D'Haro, Luis Fernando, Cordoba, R., San-Segundo, R., Ferreiros, J., Pardo, J. M. "Design and evaluation of acceleration strategies for speeding up the development of dialog applications" *Speech Communication*, Vol. 53, Iss. 8, pp. 1002-1025, ISSN: 0167-6393, 2011.
[28] Jiang, Ridong, Rafael E. Banchs, Seokhwan Kim, Luis F. D'Haro, Andreea I. Niculescu, and Kheng Hui Yeo. "Configuration of dialogue agent with multiple knowledge sources." *In Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, 2015 Asia-Pacific, pp. 840-849. IEEE, 2015.
[29] Jiang, Ridong, Yeow Kee Tan, Dilip Kumar Limbu, A. T. Tung, and Haizhou Li. "A configurable dialogue platform for ASORO robots." *In Asia Pacific Signal and Information Processing Association Annual Summit and Conference, APSIPA* ASC. 2011.
[30] Niculescu, Andreea I and Banchs, Rafael E. "Strategies to cope with errors in human-machine spoken interactions: using chatbots as back-off mechanism for task-oriented dialogues" *in Proceedings ERRARE 2015 - Errors by Humans and Machines in multimedia, multimodal and multilingual data processing*, 2015.