FAIM/ISCA Workshop on Artificial Intelligence for Multimodal Human Robot Interaction
14-15 July 2018, Stockholm, Sweden

Human-Robot Interaction Requires More Than Slot Filling – Multi-Threaded Dialogue for Collaborative Tasks and Social Conversation*

Ioannis Papaioannou[1], Christian Dondrup[1], Oliver Lemon[1]

***Abstract*— Work on spoken dialogue systems (SDS) has largely been dominated by "slot filling" applications for the past decade or more, where information-gathering tasks such as restaurant search and flight booking have been the main focus of research. An important class of dialogues, about planning and executing tasks, has correspondingly been somewhat neglected. However, planning and execution dialogues become very important when we consider Human-Robot Interaction (HRI), as does the ability to carry out "social" open-domain conversation. This paper describes a new architecture that supports complex multi-threaded task planning and execution dialogues, interleaved with "social" dialogue with an open-domain chatbot.**

I. INTRODUCTION

Spoken language is the most efficient and natural method of human-human communication. Consequently, it provides an excellent means of communication between human users and artificial agents [4]. Hence, a large amount of research has been devoted to implementing task-based conversational agents (e.g. [7], [14]), though most of this research is focused only on successfully activating single tasks which do not have a duration (like for example in assistants such as Google Assistant or Siri), and less on monitoring task execution and dealing with multiple ongoing tasks within a natural flow of dialogue and interaction. Such issues are more important for robots, since tasks generally take time to execute, can interrupt each other, and can fail for various reasons, or be suspended and later resumed. In addition, multiple tasks can be planned for future execution, either in series, nested, or in parallel.

In addition to a requirement to support dialogue about multiple ongoing tasks, future plans, and ongoing task execution status, in recent years it has become recognised that "social chat" is also an important element of more engaging and usable dialogue interfaces, especially when having the option of an *open domain* dialogue with the user, like in the work presented in [5], [6]. As shown in [11], [12], adding social chat to a task-based agent not only has *no* negative impact on the system's task completion performance, but it is perceived as more pleasant and as meeting the users' expectations better than a purely task based system.

Addressing these issues and inspired by and building on the ability to manage multiple simultaneously executed tasks using Natural Language as introduced in previous work [9], we propose an extension to the premise of our recent experiment [11] by combining this ability of multi-threaded task management with a social dialogue component using the *Alana* open-domain conversational system presented in [10]. The proposed framework, therefore, improves upon the ideas presented in [9] by allowing the system to build rapport with the user through social interaction as shown in [11] and refines the system architecture by combining the two dialogue systems, i.e. task-based and social chat, into one overall dialogue management system as part of *Alana* [10], which allows for a more coherent interaction.

The system is designed for the use on a Pepper robot – using the Robot Operating System (ROS) – which will be deployed in a shopping mall in Finland, as part of the EC project MuMMER, where it will interact with customers to provide them with information such as the location of and route to certain shops, current offers, and where to find certain products while at the same time being able to engage in small talk using open-domain dialogue.

II. METHODOLOGY

The proposed system combines the dialogue management concepts of [9], [10] with a planning and execution framework [2] using an Arbitration component (see Fig. 1) which handles the communication between the two and decides whether to send the output of *Alana* to the Planner, the Knowledge-Base (KB), or to the TTS. For example, user requests and planner responses or KB queries are always directly sent to Alana but Alana's responses might be sent to the planner or the TTS. The Arbiter also keeps track of the currently running plans/tasks and sends results of KB queries that require user input to the task that required this information. For example, dialogue may be needed to disambiguate between several possible shops that sell a certain product before generating a route description to either of them. In the following, we discuss the need for multi-threaded dialogue in Human-Robot Interaction (HRI) and describe the two systems that are combined via the Arbiter node to achieve it, i.e. the planning and execution framework, and dialogue management.

A. Multi-threaded dialogue

Each task that a user could ask the robot to execute triggers the generation of a plan – which might require several actions and nested sub-actions to be executed in sequence

[1]I. Papaioannou, C. Dondrup, and O. Lemon are with the School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, Scotland, UK {i.papaioannou, c.dondrup, o.lemon} at hw.ac.uk

10.21437/AI-MHRI.2018-15

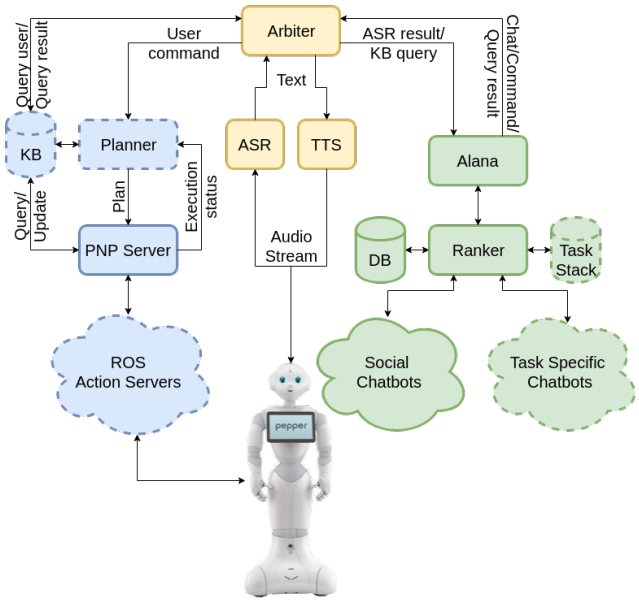

Fig. 1. Proposed architecture of the system under development showing the planning and execution framework on the left (blue), the dialogue management on the right (green), and the components combining the two systems in the centre (yellow). The components with a dashed outline are currently under development whereas the components with a solid outline are already implemented and are being tested on the robot.

to fulfil the desired goal. These actions could also require intermediate input from the user to disambiguate or clarify certain requests, and the user might want to pause or cancel the whole plan at any point in time. As a result, in order to successfully complete a task more than one dialogue turn might be required, and the question arising is how to build a system that is robust, and allows the user to not only interleave several tasks as in [9] but also to engage in social chat while performing the tasks at hand, as shown in the example dialogue (Fig. 2).

USER: Where can I find a coffee shop?
SYSTEM: There are two close by. Do you prefer Shop_1 or Shop_2?
USER: Oh, I never asked for your name! What are you called?
SYSTEM: My name is Alana! Nice to meet you. So, about the coffee shop. Which one would you prefer?
USER: Cool! Anyway I prefer Shop_2.
SYSTEM: Alright! Directions to Shop_2. Do you see the red sign on the left? ...

Fig. 2. Example dialogue

In this example, the user starts with a clear intent of wanting to find a coffee shop, but they deviated from what the system was expecting in order to complete the task. After a couple of turns of social dialogue, the previous system's request is re-raised and then answered, allowing it to continue the currently active task. To achieve that, we will have to handle multi-threaded dialogue, keep track of unfinished tasks, include the ability to pause and resume tasks, and re-raise previous questions to prompt the user. There has been promising work in this field [3], [7], [8], but our main focus is to implement a complete and scalable system that produces a more natural and human-like flow of conversation while switching between several task threads and non-task dialogue. Based on the work in [8], every time a new user utterance relates to a task which the robot is able to fulfil, the system first checks whether it can be attached to a preexisting task thread, or else it will start a new thread. All the tasks are being queued in a *Task Stack* in order of recency, so that once a task is completed, it pops off the stack. The Task Stack keeps detailed status information for each task, like unmet preconditions, questions to the user that need to be re-raised un-answered, etc. Some tasks in the stack are also prioritised and can bypass the recency order, in the sense that some tasks should be considered more urgent and must be performed first (e.g. the robot is running out of battery, robot has tipped over and requires human help, etc.).

Another advantage of this approach is that it allows the user to interrupt an active task even while the robot is executing it (e.g. with a command "stop that"). In this example, *Alana*'s co-reference resolution submodule will resolve *"that"* into a meaningful task identifier, and send an interrupt command to the Arbiter node which, in turn, will interrupt the corresponding task. More detailed statuses can also be provided in the same way (e.g. "What are you doing now?") as described in [9], where the system will generate a coherent and natural utterance to the user, preferably "echoing" the user's language (e.g. "I am directing you to Shop_2" assuming an interaction such as in Fig. 2).

B. Planning and Execution Framework

To allow for multi-threaded task execution and dialogue, the approach to task planning and execution has to be flexible, pausable, and multi-threaded. The approach here is largely based on the system presented in [2]. This system is comprised of a planning component [1] using PDDL which produces sequences of actions – based on the current state of the world – that need to be carried out in order to fulfil a given goal and an execution component to execute these actions. To this end, the resulting plan is transformed into a Petri-Net Plan (PNP) [15] which allows for the addition of recovery behaviours automatically generated from the preconditions and effects of the actions and the concurrent execution of multiple actions. This system has already proved to be flexible and robust enough for social interaction in the target domain. However, to allow for the required type of interaction, it will be enriched with the ability to execute several PNPs simultaneously, each being executed independently and concurrently.

Moreover, to eliminate one of the biggest problems impacting the fluency of the interaction between user and robot, i.e. delays arising from (re-)planning, the system architecture described here uses the principle of *recipes and resources* as developed in [9] instead of a PDDL-based world definition

as in [2]. These recipes, which describe the tasks involved in achieving a given goal and the resources required to execute these tasks, e.g. controlling parts of the robot like the head for gazing, will be transformed into PNPs if and only if the resources are currently available (as stored in the KB) and can then be concurrently executed together with other plans that might already be running. Using the ability of the PNPs to include precondition and effect checks together with recovery behaviours, it is trivial to skip redundant actions whose effects have already been achieved or define repair actions in case an action was unsuccessful and, therefore, should eliminate the need for constant re-planning.

In cases where an action requires knowledge which is unknown to the robot and has not been provided by the user, e.g. which shop they actually want to go to to buy a coffee (cf. Fig. 2), it queries the KB asking for a coffee shop. Given that there are multiple possible targets, the KB tries to disambiguate by formulating a query to the user via the Arbiter node and *Alana* listing the options to chose from, i.e. Shop_1 and Shop_2 in the example. As described above, the Arbiter will identify a possible reply coming from the user via *Alana* as referring to a certain plan and action and forward the result to the KB which will then be able to answer the query of the action. In the meantime, the PNP will be paused and wait for a reply. Hence, in order to trigger the execution of a certain plan, to answer KB queries, or to inform about, change, or cancel ongoing tasks, the planner and the KB use the Arbiter node mentioned above to query the user via the *Alana* system which is described in the following.

C. Dialogue Management

Alana is a scalable, highly customizable open-domain dialogue system comprised of several interchangeable components (see Fig. 1), combined into 4 basic modules: a *Natural Language Understanding (NLU) / Preprocessing* module, a *Post-processing* module, a *Ranker* module which selects the most appropriate response and an *ensemble of sub-bots*, with each sub-bot designed to provide different response candidates. When the user speaks to the robot, the NLU module extracts the intent of the user, as well as several other meta-data, e.g. noun phrases, named entities, etc., from the utterance which are then forwarded to the ensemble of sub-bots. Each bot produces one (or more) possible response(s) based on the type and functionality of the bot. Some sub-bots produce responses about various subjects, attempting to increase social engagement (e.g. information retrieval from wikipedia, news, or ELIZA-style responses [13]), where others perform tasks as described above. These responses are then gathered and based on the user intent, the current dialogue context, and the dialogue history, the best candidate is selected by the *Ranker*. This non-deterministic selection of an appropriate response helps to keep a natural and contextually appropriate flow of conversation which is not repetitive or predictable.

As described in section II-B, *Alana* communicates with the robot using the Arbiter node. As such, the sub-bots in the ensemble that are responsible for generating task-related dialogue responses, i.e. the *Task Specific Chatbots* in Fig. 1, exchange information with the planning and execution framework via the inclusion of specifically formatted commands in the response which, besides text to be synthesised via the TTS, hold all the information required for the next task dialogue act (as opposed to the social dialogue sub-bots that require and respond using text messages only). Using this format of response, *Alana* is agnostic regarding whether the produced response includes planner commands and, therefore, by treating planner commands and simple responses equally inside of *Alana*, all the task related information is retained within the shared dialogue context in the DB (cf. Fig. 1), free to be used by any other bot in the ensemble. For example, a user that asks where a coffee shop is in the mall (interaction with i.e. bot A), might later during the interaction hear a fun fact about coffee (from bot B), or get a voucher for a free coffee (from bot C). Similar to the execution plan for actions described in II-B, the possible interactions between task sub-bots and user are described in recipes which hold all possible interactions that might arise during the execution of a task such disambiguations, clarifications, failures, success, etc. The use of task-specific recipes for the dialogue allows, therefore, scalability of the system by defining task specific templates for user interaction and dealing with all possible requests and statuses the system might produce.

III. DISCUSSION

Having an embodied dialogue system such as a robot in HRI creates the need to be able to combine task execution and social chat as shown in previous work [11], [12]. Since robots are physical entities, executing a task might take time and/or several dialogue turns. In the meantime, the user might try to chat to the robot or request different tasks. In this paper, we present an approach for dealing with this highly complex multi-threaded type of interaction by proposing a tightly integrated system which relies on the strength of both the planning and execution framework and the dialogue system described above. The use of recipes describing on the one hand the sequence of actions, including their preconditions and effects, which are required to achieve the given goal and on the other hand the dialogue acts that might arise during the execution of the task to deal with ambiguous statements, clarify uncertainties, or deal with errors, creates a scalable and extendable system that is able to handle all possible variations of the task. Combined with the open-domain dialogue features of *Alana*, the system provides a contextually natural and more fluent interaction with the user.

ACKNOWLEDGMENTS

*The research leading to these results has received funding from the European Commissions H2020 programme under grant agreement No. 688147, MuMMER project.

REFERENCES

[1] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtós, and M. Carreras. Rosplan: Planning in the robot operating system. In ICAPS, pages 333–341, 2015.

[2] C. Dondrup, I. Papaioannou, J. Novikova, and O. Lemon. Introducing a ros based planning and execution framework for human-robot interaction. In Proceedings of the 1st ACM SIGCHI International Workshop on Investigating Social Interactions with Artificial Agents, ISIAA 2017, pages 27–28, New York, NY, USA, 2017. ACM.

[3] P. A. Heeman, F. Yang, A. L. Kun, and A. Shyrokov. Conventions in human-human multi-threaded dialogues: A preliminary study. In Proceedings of the 10th International Conference on Intelligent User Interfaces, IUI '05, pages 293–295, New York, NY, USA, 2005. ACM.

[4] C. R. Huyck. Dialogue based interfaces for universal access. Universal Access in the Information Society, 10(3):267–274, Aug 2011.

[5] K. Jokinen and G. Wilcock. Open-domain information access with talking robots. In Proceedings of the SIGDIAL 2013 Conference, pages 360–362, 2013.

[6] G. W. Kristiina Jokinen. Wikitalk: A spoken wikipedia-based open-domain knowledge access system. In 24th International Conference on Computational Linguistics, page 57. Citeseer, 2012.

[7] A. L. Kun, A. Shyrokov, and P. A. Heeman. Spoken tasks for human-human experiments: Towards in-car speech user interfaces for multi-threaded dialogue. In Proceedings of the 2nd International Conference on Automotive User Interfaces and Interactive Vehicular Applications, AutomotiveUI '10, pages 57–63, New York, NY, USA, 2010. ACM.

[8] O. Lemon and A. Gruenstein. Multithreaded context for robust conversational interfaces: Context-sensitive speech recognition and interpretation of corrective fragments. ACM Trans. Comput.-Hum. Interact., 11(3):241–267, Sept. 2004.

[9] O. Lemon, A. Gruenstein, A. Battle, and S. Peters. Multi-tasking and collaborative activities in dialogue systems. In Proceedings of the 3rd SIGdial Workshop on Discourse and Dialogue - Volume 2, SIGDIAL '02, pages 113–124, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

[10] I. Papaioannou, A. Cercas Curry, J. L. Part, I. Shalyminov, X. Xu, Y. Yu, O. Duek, V. Rieser, and O. Lemon. Alana: Social dialogue using an ensemble model and a ranker trained on user feedback. In Proc. AWS re:INVENT, 2017.

[11] I. Papaioannou, C. Dondrup, J. Novikova, and O. Lemon. Hybrid chat and task dialogue for more engaging hri using reinforcement learning. In Proc. RO-MAN. IEEE, 2017.

[12] I. Papaioannou and O. Lemon. Combining chat and task-based multimodal dialogue for more engaging hri: A scalable method using reinforcement learning. In Proc. HRI 17 Companion, 2017.

[13] J. Weizenbaum. ELIZA – A Computer Program For the Study of Natural Language Communication Between Man and Machine. Communications of the ACM, 9(1):36–35, 1966.

[14] T. Wen, M. Gasic, N. Mrksic, L. M. Rojas-Barahona, P. Su, S. Ultes, D. Vandyke, and S. J. Young. A network-based end-to-end trainable task-oriented dialogue system. CoRR, abs/1604.04562, 2016.

[15] V. A. Ziparo, L. Iocchi, P. U. Lima, D. Nardi, and P. F. Palamara. Petri net plans. Autonomous Agents and Multi-Agent Systems, 23(3):344–383, 2011.