Conversational Interface for Managing Non-trivial Internet-of-Things Systems

André Sousa Lago[1], João Pedro Dias[1,2(✉)], and Hugo Sereno Ferreira[1,2]

[1] DEI, Faculty of Engineering, University of Porto, Porto, Portugal
{up201303313,jpmdias,hugosf}@fe.up.pt
[2] INESC TEC, Porto, Portugal

Abstract. Internet-of-Things has reshaped the way people interact with their surroundings. In a smart home, controlling the lights is as simple as speaking to a conversational assistant since everything is now Internet-connected. But despite their pervasiveness, most of the existent IoT systems provide limited out-of-the-box customization capabilities. Several solutions try to attain this issue leveraging end-user programming features that allow users to define rules to their systems, at the cost of discarding the easiness of voice interaction. However, as the number of devices increases, along with the number of household members, the complexity of managing such systems becomes a problem, including finding out why something has happened. In this work we present Jarvis, a conversational interface to manage IoT systems that attempts to address these issues by allowing users to specify time-based rules, use contextual awareness for more natural interactions, provide event management and support causality queries. A proof-of-concept was used to carry out a quasi-experiment with non-technical participants that provides evidence that such approach is intuitive enough to be used by common end-users.

Keywords: Internet-of-Things · Conversational assistants · Software engineering · Natural Language Processing · Visual programming

1 Introduction

The Internet of Things (IoT) is usually defined as the networked connection of everyday objects with actuating and sensing capabilities, often equipped with a collective sense of intelligence [21]. The integration of such objects creates a vast array of distributed systems that can interact with both the environment and the human beings around them, in a lot of different ways [21]. This flexibility of IoT systems has enabled their use across many different product areas and markets, including smart homes, smart cities, healthcare, transportation, retail, wearables, agriculture and industry [17].

Still, one of the most visible application of IoT are *customized smart spaces*, such as *smart homes* as the current technology make it possible for consumers to create a customized IoT experience based on *off-the-shelf* products [16].

V. V. Krzhizhanovskaya et al. (Eds.): ICCS 2020, LNCS 12141, pp. 384–397, 2020.
https://doi.org/10.1007/978-3-030-50426-7_29

The initial popularity of devices such as single-board computers and low-cost micro-controllers, followed by widespread cloud-based solutions controlled by mobile phones, it is now commonplace to remotely interact with a myriad of devices to perform automated tasks such as turning the lights on and opening the garage door just before one arrives home [16,22]. But as the number of devices and interactions grows, so does the management needs of the system as a whole, as it becomes essential to understand and modify the way they (co)operate. In the literature this capability commonly known as *end-user programming* [6], and once we discard trained system integrators and developers, two common approaches emerge: visual programming tools and conversational assistants [22].

Visual programming solutions are usually deployed as centralized orchestrators, with access to the devices and components that comprise such systems. These platforms range from simplified *if-then* rules (*e.g.* IFTTT[1]) to exhaustive graphical interfaces (*e.g.* Node-RED[2]) through which one can visualize, configure and customize the devices and systems' behaviour [7,19,20]. Most visual approaches attempt to offer integration with third-party components (*e.g.*, google calendar), so that their services can be used as part of the system's behavioural rules.

These solutions, however, possess some disadvantages for non-technical *end-users*. Consider a Node-RED system orchestrating an user's smart home with multiple devices. Even in situations where there are only a couple of dozen rules defined, it can be challenging to understand why a specific event took place due to the overwhelming data flow that results from these. Furthermore, just one dozen rules can already lead to a system not possible to visualize in a single screen [11]. The more rules one adds, the harder it becomes to conceptually grasp what the system can do. Part of the reason is because they are built to be *imperative*, not *informative*; current solutions mostly lack in meta-facilities that enable the user or the system to *query* itself [5].

Another common, and sometimes complementary, alternative to visual programming, is the many conversational assistants in the market, such as Google Assistant, Alexa, Siri and Cortana, that are capable of answering natural language questions and which recently gained the ability to interact with IoT devices (see [18] and [15] for a comparison of these tools). Amongst the most common features they provide is allowing direct interaction with sensing and actuating devices, which enables the *end-user* to *talk* to their light bulbs, thermostats, sound systems, and even third-party services. The problem with these solutions is that they are mostly comprised of *simple* commands and queries directly to the smart devices (*e.g. is the baby monitor on?*", "*what is the temperature in the living room?*", or "*turn on the coffee machine*". These limitations mean that although these assistants do provide a comfortable *interaction* with devices, a huge gap is easily observable regarding their capabilities on *managing* a system as a whole and allowing the definition of rules for how these *smart spaces* oper-

[1] IFTTT, https://ifttt.com.
[2] Node-RED, https://nodered.org.

ate. Even simple rules like "*close the windows everyday at 8pm*" or "*turn on the porch light whenever it rains*" are currently not possible, unless one manually defines every single one of them as a capability via a non-conversational mechanism. Furthermore, most assistant are deliberately locked to specific vendor devices, thus limiting the overall experience and integration.

One can conclude that although current smart assistants can be beneficial and comfortable to use, they do not yet have the complexity and completeness that other systems like Node-RED. Meanwhile, visual programming environments are still far too technical for the common *end user*. In this paper, we propose a system that tackles the problem of *managing* IoT systems in a conversational approach, towards shortening the existing feature gap between assistants and visual programming environments. Parts of this work are from [13] master's thesis.

The rest of this document is structured as follows: Sect. 2 provides a summary of related works which identify open research challenges; in Sect. 3 we propose our approach to support *complex* queries in conversational assistants, which implementation details are further presented in Sect. 4; we proceed in Sect. 5 to evaluate our approach using simulated scenarios and experimental studies. Finally, Sect. 6 drafts some closing remarks.

2 Related Work

There exists some work in this area that recognize the problem of controlling and managing IoT infrastructures by an *end-user* via a several approaches.

Kodali et al. [12] present an home automation system to *"increase the comfort and quality of life"*, by developing an Android app that is able to control and monitor home appliances using MQTT, Node-RED, IFTTT, Mongoose OS and Google Assistant. Their limitations lie in that the *flows* must first have been first created in Node-RED, and the conversational interface is used just to trigger them, ignoring all the *management* activities.

Austerjost et al. [3] recognized the usefulness of voice assistants in home automation and developed a system that targets laboratories. Possible applications reported in their paper include stepwise reading of standard operating procedures and recipes, recitation of chemical substance or reaction parameters to a control, and readout of laboratory devices and sensors. As with the other works presented, their voice user interface only allows controlling devices and reading out specific device data.

He et al. [9], concludes that, even with conversational assistants, most of IoT systems have usability issues when faced with complex situations. As example, the complexity of managing devices schedules rises with the number of devices and the common conflicting preferences of household members. Nonetheless, as concluded by Ammari et al. [2], controlling IoT devices is one of the most common uses of such assistants.

Agadakos et al. [1] focus on the challenge of understanding the causes and effects of an action to infer a potential sequence. Their work is based on a mapping the IoT system' devices and potential interactions, measuring expected

behaviours with traffic analysis and side-channel information (e.g. power) and detecting causality by matching the mapping with the collected operational data. This approach would potentially allow the *end user* to ask *why is something happening*, at the cost of modified hardware and a convoluted side-channel analysis. They did not attempted to port their findings into a conversational approach.

Braines et al. [4] present an approach based on Controlled Natural Language (CNL) – natural language using only a restricted set of grammar rules and vocabulary – to control a smart home. Their solution supports (1) *direct question/answer exchanges*, (2) *questions that require a rationale as response* such as *"Why is the room cold?"* and (3) *explicit requests to change a particular state*. The most novel part of their solution is in trying to answer *questions that require a rational response*, however they depend on a pre-defined smart home model that maps all the possible causes to effects.

From the above analysis, the authors were not able to found any solution that would simultaneously provide: (1) a non-trivial management of an IoT system, (2) be comfortable and easy to use by a non-technical audience, and (3) allow the user to better understand how the system is functioning. By *non-trivial* we mean that it should be possible to define new rules and modify them via a conversational approach, achieving a *de facto* integration of multiple devices; not just directly interacting with its basic capabilities. The comfort would be for the user not to have to move or touch a device to get his tasks done (i.e. using voice), or edit a Node-RED visual flow. As to understanding their system's functioning, we mean the ability to grasp *how* and *why* something is happening in their smart space. This last point, combined with the other two, would ideally allow someone to simply ask why something happens.

3 Solution Overview

We propose the development of a conversational bot dedicated to the management of IoT systems that is capable of defining and managing complex system rules. Our prototype is called **Jarvis**, and is available as a reproducible package [14].

Jarvis's abilities reach across different levels of operational complexity, ranging from direct one-time actions (e.g. *turn on the light*) to repeating conditional actions (e.g. *when it is raining, close the windows*). Jarvis also lets the user easily *understand* and *modify* the rules and cooperation of multiple devices in the system, through queries like *why did the toaster turn on?* In these cases, we incorporated Jarvis with *conversational awareness* to allow for chained commands; the following dialogue exemplifies this particular feature:

User: "*Why did the toaster turn on?*"
Jarvis: "*You told me to turn it on at 8 AM.*"
User: "*Okay, change it to 7:50 AM.*"
Jarvis: "*Sure, toaster timer was changed.*"

... the reader would not make sense on its own. We believe that such features improve the user's experience since it avoids

repeating information that has already been mentioned in the conversation, and presents a more *natural* (conversational) interaction.

To ease the integration with nowadays systems and provide us an *experimental reproducible environment*, we integrated the interface with existing platforms such as Google Assistant[3] and Slack[4], amongst others. We made sure to provide the ability for Jarvis to interact both via *voice* and *text*.

4 Implementation Details

Figure 1 presents the high-level software components of Jarvis. Each component and corresponding techniques are explained in the following subsections.

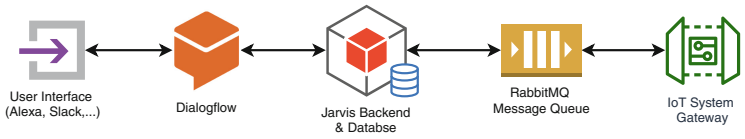

Fig. 1. Jarvis overall architectural components.

4.1 Conversational Interface

To develop the conversational interface, we decided to opt for Dialogflow[5] as this platform provides built-in integration with multiple popular *frontends* and there exists extensive documentation for this purpose [10]. In this case, we used (1) the Slack team-communication tool, and (2) Google Assistant, so that both text and voice interfaces were covered. In the case of Google Assistant, the user may use any supported device paired with their account to communicate with Jarvis, following a known query prefix such as *"Hey Google, talk to Jarvis"*. Regardless of which type of interface is used, the result is converted to *strings* representing the exact user query and subsequently sent to Dialogflow's backend (thus overcoming potential challenges due to Speech Recognition), which are then analyzed using Natural Language Processing (NLP) techniques. Advancement of the existent NLP techniques made available by Dialogflow falls out-of-the-scope of this work.

4.2 Dialogflow Backend

Upon receiving a request, Dialogflow can either produce an automatic response or send the parsed request to a fulfillment *backend*. This component is thus responsible for parsing the incoming *strings* into a *machine understandable* format (JSON). There are a few key concepts that are leveraged in our implementation:

[3] Google Assistant, GoogleAssistant.
[4] Slack, https://slack.com.
[5] Dialogflow, https://dialogflow.com/.

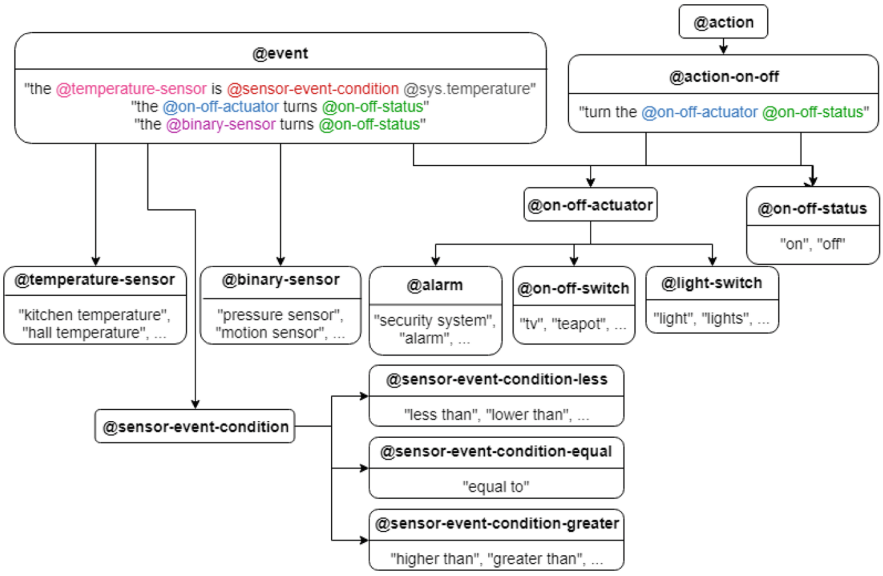

Fig. 2. Main entities defined in Jarvis' Dialogflow project.

Entity. Things that exist in a specific IoT ecosystem can be represented by different literal strings; for example, an entity identified by `toggleable-device` may be represented by *"living room light"* or *"kitchen light"*. Additionally, entities may be represented by *other* entities. Dialogflow use of the `@` symbol (*i.e.* `@device`) for entities, and provides some system's defaults;

Intent. An intent represents certain type of user interaction. For instance, an intent named *Turn on/off device* may be represented by `turn the @device on` and `turn the @device off`. For a request such as *"turn the kitchen light on"*, Dialogflow understands that `@device` corresponds to *kitchen light* and provides that data to the fulfillment backend;

Context. Contexts allow intents to depend on previous requests, enabling the creation of context-aware interactions. These are what supports queries such as *"cancel that"* or *"change it to 8AM"*.

Multiple *intents*, *entities* and *contexts* were defined in Jarvis and the main ones are illustrated in Fig. 2. Here we provide in detail one of its *intents*:

Event Intent

Usage Creates an action that is performed upon a certain event, such as an activity of another device or a change of a device's status.
Definition `@action:action when @event:event`
Example *Turn the bedroom light on when the living room light turns off*

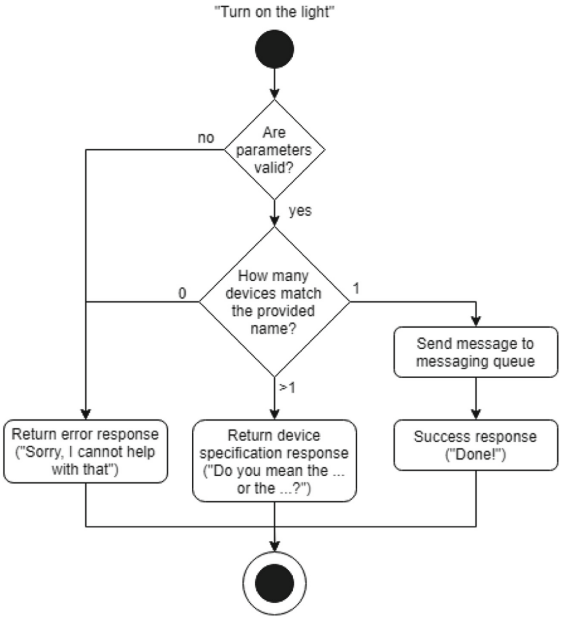

Fig. 3. Activity diagram for the parsing of the query *turn on the light*.

With the above definitions, this component takes requests and builds the corresponding objects containing all actionable information to be sent to the Jarvis backend for further processing.

4.3 Jarvis Backend

For each of the defined intents, this component has an equivalent class responsible for (a) parsing the request, (b) validating its request parameters (*e.g.* device name or desired action), and (c) generating an appropriate response. An overview is provided in Fig. 3. Should the request contain errors, an *explanatory* response is returned. When all the parameters are considered valid, but the intended device is *unclear* (*e.g.* user wants to turn on the light, but there is more than one light), the generated response specifically asks the user for further clarification in order to gain *context*. To tackle cancellation intents, we model all actions using the COMMAND design pattern, thus providing both a straightforward *execute* and *undo* mechanism, as well as an history of performed actions. For most intents, such as *direct actions* or *"why did something happen?"* queries, the effects are immediate. However, *period actions*, *events* and *causality queries* require a different design approach so that they can perform actions on the backend without the need for a request to trigger them.

Period Actions and Events. A *period action* is an intent be carried and then undone after a certain period (*e.g. "turn on the light from 4 pm to 5 pm"*). In these scenarios, we generate a *state machine* to differentiate between all the different action status, such as (a) nothing has executed yet (before 4PM), (b) only the first action was executed (after 4 but before 5PM), and (c) both have been executed (after 5PM). We use a combination of *schedulers* and *threads* to guarantee proper action, and abstract all these details inside the COMMAND pattern. The same strategy applies for rules such as *"turn on the light every day at 5 pm"*, with the appropriate state machine and scheduler modifications. This mechanism is (obviously) different for events that are the result of intentions such as *"turn on the kitchen light when the presence sensor is activated"*. Here, we leverage a *publish-subscribe* approach by orchestrating multiple unique and identifiable *message queues* for the devices' actions and state transitions. Upon startup of the backend, we create class listeners that subscribe the corresponding event queues of available devices, and callback the Jarvis backend when appropriate. This orchestration management is dynamic, and depends on the specific rules that are defined. In the aforementioned intent, we would add an observer to look for messages on the presence sensor's event queue with the value `on`.

Causality Queries. This relate to the user asking why something happened (*e.g. "why did the light turn on?"*). To implement them, we augment each COMMAND to determine whether it can cause a specific condition to be true. This *per se* solves some scenarios where the answer can be found by looking at the history of executed commands (*e.g. "because you asked me to turn it on at 3:56PM"*). However, there might exist multiple rules may have cause the condition to be true, in which case it is not enough to blame the latest logged command. Instead, there are two possible approaches: (a) return the earliest possible cause, or (b) use a heuristic to infer the most relevant one. Another non-trivial scenario is where the explanation is due to a chain of interconnected rules. Here, it seems that one can (a) reply with the complete chain of events, (b) reply with the latest possible cause, or (c) engage in a *conversation* through which the user can explore the full chain of events as they deem adequate (*e.g. "tell me more about things that are triggered by rain"*). In this work, we opted to use the earliest possible cause for the first scenario, and the latest for the second; more complex alternatives can be found in [1,4].

5 Experiments and Results

In order to understand how Jarvis compares to other systems, we established a baseline based on (1) a visual programming language, and (2) a conversational interface. Node-RED was picked amongst the available visual programming solution, as it is one of the most popular visual programming solutions [8]. It follows a flow-based programming paradigm, providing its users with a web-based application through which they can manage rules via *connections* between *nodes* that represent devices, events and actions [20]. Google Assistant was selected for the

Table 1. Simulated scenarios comparison.

Scenario	Jarvis	Google Assistant	Node-RED
One-time action	•	•	•
One-time action w/unclear device	•	·	·
Delayed action	•	·	•
Period action	•	·	•
Daily repeating action	•	·	•
Daily repeating period action	•	·	•
Cancel last command	•	·	·
Event rule	•	·	·
Rules defined for device	•	·	·
Causality query	•	·	·

conversational interface due to its naturality[6]. There are plenty of ways users can interact with it: (a) the standalone Google apps, (b) built-in integration with Android and Chrome OS, or (c) with standalone hardware such as the Google Home. We compare to this baseline according to two criteria: (1) the *number of different features*, and (2) their *user experience* in terms of easiness of usage and intuitiveness. For the first, we created a list of simulated scenarios to assess the ability to manage IoT systems. We then performed a (quasi-)controlled experiment with users to assess the second criteria.

5.1 Simulated Scenarios

Table 1 summarizes the comparison of our prototype to the chosen baseline. It is important to clarify that *one-time action w/ unclear device* refers to actions like *"turn on the light"* with which Jarvis asks the user to clarify which device he means based through responses such as *"do you mean the bedroom or living room light?"*. A *cancel last command* refers to the ability of undoing the last action or rule creation by specifically saying that. Finally, *rules defined for device* refers to the user performing queries that require introspection, such as *"what rules are defined for the bedroom light?"*

It is easy to observe that our prototype provides several features that are not present in either the Google Assistant or Node-RED. Both of these products do a lot more than these features, but especially with the Assistant, the advantage is clear since the only kind of IoT feature it supports is the *one-time action*. Our second conclusion is that it is possible to bring some of the features currently available in visual programming environments to a conversational interface; the converse (how to bring conversational features to Node-RED), eludes the authors.

[6] The work by López et al. [15] compares Alexa, Google Assistant, Siri and others, and claim that although "*Siri was the most correct device (...) Google assistant was the one with the most natural responses*".

5.2 (Quasi-)Controlled Experiment

We performed a (quasi-)controlled experiment with 17 participants, to gain insight into how *end users* responded to a conversational approach. Our sample includes mostly participants without formal technological skills (14), with ages ranging from 18 to 51. We made sure that (a) all were familiar with basic technologies, though, such as basic usage of smartphones and the Internet, and (b) that even non-native English participants had adequate speaking and understanding skills.

Methodology. Each participant was given 5 tasks (1 control task and 4 study tasks) to be performed with the help of Jarvis, using Google Assistant as the system interface. As an example, this was one of the sets of tasks given to participants within a scenario with a *living room light*, *bedroom light* and a *living room motion sensor*:

- **Task 0 (control):** *Turn on the living room light*;
- **Task 1:** *Turn the living room light on in 5 min*;
- **Task 2:** *Turn the living room light on when the motion sensor triggers*;
- **Task 3:** *Check the current rules defined for the bedroom light, and then make it turn on everyday at 10pm*;
- **Task 4:** *Find out the reason why the bedroom light turned on. Ask Jarvis why it happened and decide whether the answer was explanatory.*

The only instructions given were that they should talk to the phone in a way that feels the most natural to them to complete the task at hand. Besides the tasks, participants were also given the list of IoT devices available in the simulated smart house that they would be attempting to manage through. As a way of increasing the diversity and reducing the bias of the study, we created two different sets of tasks that participants were assigned to randomly. Each set also had different devices, with different smart house topologies. The participants were assigned to one of the test sets randomly.

Variable Identification. For each of the tasks, we collected (1) if the participant was able to complete it, (2) the time taken to complete, and (3) the number of unsuccessful queries. This count was made separately for (a) queries that were not understood by the assistant's speech recognition capabilities (*e.g.* microphone malfunction, background noise), (b) queries where the user missed the intention or made a syntactic/semantic error (*e.g. "turn up the lighting"*), and (c) valid queries that an human could interpret, but that Jarvis was unable to.

Subjective Perception. After completing the tasks, we introduced a non-conversational alternative (Node-RED), explaining how all tasks could have been performed using that tool. We inquired the participants whether they perceived any advantages of Jarvis over such a tool, and whether they would prefer Jarvis over non-conversational tools. Finally the participants were asked if they had any suggestions to improve Jarvis and the way it handles system management.

Table 2. Experimental results (task completion rate, task time and incorrect queries), including average and median values.

		Time (s)		IQ (Ast)		IQ (User)		IQ (Jvs)		IQ	
Task	Done	Avg	Med	Avg	Med	Avg	Med	Avg	Med	Avg	Med
0 (1)	94%	6.40	6.0	0.13	0.0	0.25	0.0	0.13	0.0	0.50	0.0
1 (1)	94%	7.10	7.0	0.38	0.0	0.50	0.5	0.00	0.0	0.50	0.5
2 (1)	88%	10.00	10.0	0.75	0.5	0.63	0.5	0.25	0.0	1.00	1.0
3 (1)	100%	20.00	19.5	0.13	0.0	0.13	0.0	0.75	1.0	1.00	1.0
4 (1)	94%	9.00	8.0	0.25	0.0	0.38	0.0	0.00	0.0	0.63	0.0
0 (2)	100%	6.40	6.0	0.33	0.0	0.00	0.0	0.33	0.0	0.67	0.0
1 (2)	94%	7.60	7.0	0.11	0.0	0.00	0.0	0.44	0.0	0.56	0.0
2 (2)	100%	9.90	10.0	0.00	0.0	0.11	0.0	0.78	1.0	0.89	1.0
3 (2)	88%	19.44	19.0	0.33	0.0	0.33	0.0	0.22	0.0	0.89	1.0
4 (2)	100%	8.33	8.0	0.33	0.0	0.22	0.0	0.22	0.0	0.78	1.0

Results. Table 2 compiles the results observed during the study, each row representing a task given to the participant. Each column means:

- **Task:** number of the task (0–4) and the task set number in parenthesis (1/2);
- **Done:** percentage of participants that completed the task successfully;
- **Time:** time in seconds that participants took to complete the task;
- **IQ (Ast):** number of occurrences that queries were incorrect due to the Google Assistant not properly recognizing the user's speech;
- **IQ (User):** number of occurrences that queries were incorrect due to the user not speaking a valid query;
- **IQ (Jvs):** number of occurrences that queries were incorrect due to Jarvis not recognizing a valid query;
- **IQ:** total invalid queries, *i.e.* sum of *IQ (Ast)*, *IQ (User)* and *IQ (Jvs)*.

Discussion. The complexity of the queries increases from task 0 to task 3 since the queries require more words or interactions. This is reflected by the corresponding increase in time in both task sets. The numbers related to incorrect queries show some occurrences at the (voice) assistant level, which means the speech recognition failed to correctly translate what the participants said. Although this does not have implications on the evaluation of Jarvis, it does indicate that this sort of systems might be harder to use due if they are not truly multilingual. Directly comparing the time needed to complete a task to what would be needed to perform it in a visual programming language is meaningless; either the task is not defined, and that would require orders of magnitude longer than what we observe here, or the task is defined and the times will be obviously similar. Similarly, we also observe a few instances of incorrect queries

due to grammar mistakes or semantically meaningless, *cf. IQ (User)*, and therefore did not match the sample queries defined in Dialogflow. Nevertheless, there where grammatically incorrect user queries such as *"turn on lights"* but which still carries enough information to understand what the user's intent is. We consider more serious the number of *valid* sentences that were considered incorrect queries by Jarvis, *cf. IQ (Jvs)*. These could have been caused by either a mispronunciation of a device's name, or a sentence structure that is unrecognizable by the Dialogflow configuration. This possibly represents the most serious threat to our proposal, to which we will later dedicate some thoughts on how to mitigate it. Nonetheless, the success rate of all tasks is very high (always greater than 88%), which provides evidence that the system might be intuitive enough to be used without previous instruction or formation. These points were reflected by the participants' subjective perception, were they claimed Jarvis to be easy to use, intuitive, and comfortable; ultimately, these would be the deciding factors for end-users to prefer Jarvis over a non-conversational interface. An additional observation pertaining Jarvis' answers, particularly those regarding causality queries, were state by some users, where they claimed that if the provided response was too long, it would become harder to understand it due to the sheer increase of conveyed information. A possible solution for this problem would be to use a hybrid interface that provides both visual and audio interactions, but there could be other approaches such as an interactive dialogue that shortens the sentences.

Threats to Validity. Empirical methods seem to be one of the most appropriate techniques for assessing our approach (as it involves the analysis of human-computer interaction), but it is not without liabilities that might limit the extent to which we can assess our goals. We identify the following threats: **(1) Natural Language Capabilities,** where queries like *"enable the lights"* might not be very common or semantically correct, but it still carries enough information so that a human would understand its intention. The same happen with device identification, such as when the user says *turn on the bedroom light**s***, and the query fails due to the usage of the plural form. During our study, we observed many different valid queries that did not worked due to them not being covered by the Dialogflow configuration; **(2) Coverage error**, which refers to the mismatch between the *target* population and the *frame* population. In this scenario, our target population was (non-technical) end-users, while the frame population was all users that volunteered to participate; and **(3) Sampling errors** are also possible, given that our sample is a small subset of the target population. Repeating the experience would necessarily cover a different sample population, and likely attain different results. We mitigate these threats by providing a reproducible package [14] so other researchers can perform their own validation.

6 Conclusion

In this paper we presented a conversational interface prototype able to carry several different management tasks currently not supported by voice assistants, with

capabilities that include: (1) Delayed, periodic and repeating actions, enabling users to perform queries such as *"turn on the light in 5 min"* and *"turn on the light every day at 8 am"*; (2) The usage of contextual awareness for more natural conversations, allowing interactions that last for multiple sentences and provide a more intuitive conversation, *e.g. "what rules do I have defined for the living room light?"*; (3) Event management, that allows orchestration of multiples devices that might not necessarily know that each other exists, *e.g. "turn on the light when the motion sensor is activated"*; and (4) Causality queries, to better understand how the current system operates, *e.g. "why did the light turn on?"*

We conducted (quasi-)controlled experiments with participants that were asked to perform certain tasks with our system. The overall high success rate shows that the system is intuitive enough to be used by people without significant technological knowledge. It also shows that most challenges lie in the natural language capabilities of the system, as it is hard to predict them any user queries that have the same intrinsic meaning. We thus conclude that incorporating recent NLP advances (that were beyond the scope of this paper) would have an high impact in terms of making it more flexible to the many different ways (correct or incorrect) that users articulate the same intentions.

Nonetheless, by doing a feature comparison, we can observe that Jarvis was able to implement many features that current conversational assistants are lacking, while simultaneously being more user-friendly than the available alternatives to IoT management (such as visual programming approaches). As future work, we believe that our approach could be improved by sometimes engaging in a longer (but fragmented) conversation with the user, particularly when providing causality explanations. This would allow the user to understand more information at his own pace, but also because it would enable them to make changes to the rules as the conversation unfolds.

Acknowledgement. This work was partially funded by the Integrated Masters in Informatics and Computing Engineering of the Faculty of Engineering, University of Porto (FEUP) and by the Portuguese Foundation for Science and Technology (FCT), under research grant SFRH/BD/144612/2019.

References

1. Agadakos, I., Ciocarlie, G., Copos, B., Lepoint, T., Lindqvist, U., Locasto, M.: Butterfly effect: causality from chaos in the IoT. In: International Workshop on Security and Privacy for the Internet-of-Things, April 2018
2. Ammari, T., Kaye, J., Tsai, J.Y., Bentley, F.: Music, search, and IoT: how people (really) use voice assistants. ACM Trans. Comput.-Hum. Interact. **26**(3) (2019). https://doi.org/10.1145/3311956
3. Austerjost, J., et al.: Introducing a virtual assistant to the lab: a voice user interface for the intuitive control of laboratory instruments. SLAS TECHNOL.: Transl. Life Sci. Innov. **23**(5), 476–482 (2018)
4. Braines, D., O'Leary, N., Thomas, A., Harborne, D., Preece, A.D., Webberley, W.M.: Conversational homes: a uniform natural language approach for collaboration among humans and devices. Int. J. Adv. Intell. Syst. **10**(3/4), 223–237 (2017)

5. Dias, J.P., Faria, J.P., Ferreira, H.S.: A reactive and model-based approach for developing internet-of-things systems. In: 2018 11th International Conference on the Quality of Information and Communications Technology (QUATIC), pp. 276–281 (2018)
6. Fischer, G., Giaccardi, E., Ye, Y., Sutcliffe, A.G., Mehandjiev, N.: Meta-design: a manifesto for end-user development. Commun. ACM **47**(9), 33–37 (2004)
7. Gennari, R., Bozen-bolzano, L.U., Melonio, A., Bozen-bolzano, L.U.: End-User Development, vol. 10303. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-58735-6
8. Giang, N.K., Lea, R., Blackstock, M., Leung, V.C.: Fog at the edge: experiences building an edge computing platform. In: 2018 IEEE International Conference on Edge Computing (EDGE), pp. 9–16. IEEE (2018)
9. He, W., Martinez, J., Padhi, R., Zhang, L., Ur, B.: When smart devices are stupid: negative experiences using home smart devices. In: 2019 IEEE Security and Privacy Workshops (SPW), pp. 150–155 (2019)
10. Janarthanam, S.: Hands-on Chatbots and Conversational UI Development: Build Chatbots and Voice User Interfaces with Chatfuel, Dialogflow, Microsoft Bot Framework, Twilio, and Alexa Skills. Packt Publishing Ltd., Birmingham (2017)
11. Janssen, P., Erhan, H., Chen, K.W.: Visual dataflow modelling - some thoughts on complexity. In: Proceedings of the 32nd eCAADe Conference (2014)
12. Kishore Kodali, R., Rajanarayanan, S.C., Boppana, L., Sharma, S., Kumar, A.: Low cost smart home automation system using smart phone. In: 2019 IEEE R10 Humanitarian Technology Conference (R10-HTC)(47129), pp. 120–125 (2019)
13. Lago, A.S.: Exploring complex event management in smart-spaces through a conversation-based approach. Master's thesis, Faculty of Engineering, University of Porto (2018)
14. Lago, A.: andrelago13/jarvis: initial release, April 2020. https://doi.org/10.5281/zenodo.3741953
15. López, G., Quesada, L., Guerrero, L.A.: Alexa vs. siri vs. cortana vs. google assistant: a comparison of speech-based natural user interfaces. In: Nunes, I.L. (ed.) AHFE 2017. Advances in Intelligent Systems and Computing, vol. 592, pp. 241–250. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-60366-7_23
16. Mainetti, L., Mighali, V., Patrono, L.: An IoT-based user-centric ecosystem for heterogeneous smart home environments. In: 2015 IEEE International Conference on Communications (ICC), pp. 704–709 (2015)
17. Miranda, J., et al.: From the internet of things to the internet of people. IEEE Internet Comput. **19**(2), 40–47 (2015)
18. Mitrevski, M.: Conversational interface challenges. Developing Conversational Interfaces for iOS, pp. 217–228. Apress, Berkeley (2018). https://doi.org/10.1007/978-1-4842-3396-2_8
19. Prehofer, C., Chiarabini, L.: From IoT mashups to model-based IoT. In: W3C Workshop on the Web of Things (2013)
20. Ray, P.P.: A survey on visual programming languages in internet of things. Sci. Program. **2017**, 1–6 (2017). https://doi.org/10.1155/2017/1231430
21. Xia, F., Yang, L.T., Wang, L., Vinel, A.: Internet of things. Int. J. Commun. Syst. **25**(25) (2012)
22. Zarzycki, A.: Strategies for the integration of smart technologies into buildings and construction assemblies. In: Proceedings of eCAADe 2018 Conference, pp. 631–640 (2018)